Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет інформаційних систем та програмної інженерії
(назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

магістр
(освітній ступінь (освітньо-кваліфікаційний рівень))

на тему:

Розробка інформаційно-електронної системи для контролю відвідуваності та успішності студентів «iJournal» на основі технології JavaFX

Виконав: студент (ка) VI курсу, групи СПм-61
спеціальності (напряму підготовки)
121 Інженерія програмного забезпечення
(шифр і назва спеціальності (напряму підготовки))

(підпис) Миколюк Ю. М. (прізвище та ініціали)

Керівник (підпис) Бойко І. В. (прізвище та ініціали)

Нормоконтроль (підпис) Бойко І. В. (прізвище та ініціали)

Рецензент (підпис) Приймак М. В. (прізвище та ініціали)

м. Тернопіль – 2019

АНОТАЦІЯ

Магістерська робота містить 138 сторінок, 17 таблиць, 37 рисунків, список використаної літератури з 30 найменувань, 3 додатків.

Українська система освіти реорганізовується та показує досить хороші успіхи. Але, на сьогодні, є багато освітянських аспектів, які потребують покращення. В основному це стосується того, що учні/студенти не володіють достатньою інформацією про власну успішність та свій рейтинг, що знижує мотивацією до навчання, яка необхідна для кращого засвоєння навчального матеріалу.

Викладачі та вчителі в сучасній системі освіти витрачають багато часу на заповнення різноманітних паперових документів, зокрема журналів, замість того, щоб більше часу приділяти процесу навчання. Поточна оцінка в журналі є важливим фактором успішності учня/студента. Підсумкова/модульна оцінка в освітніх закладах розглядається як істотний фактор прогнозу успішності знань у рамках річної/семестрової оцінки по предмету/дисципліні. Саме тому систематичне виставлення оцінок у журнали необхідно, але робота із паперовими носіями потребує набагато більше часу ніж ведення електронного журналу.

Тому, впровадження електронного журналу обліку успішності та відвідуваності учнів/студентів дозволить покращити якість освітянських послуг та змінімізує корупційну складову в освіті.

Ключові слова: ОБ’ЄКТНО-ОРІЄНТОВАНА МОВА ПРОГРАМУВАННЯ, JAVA, JAVAFX, ECLIPSE, SQLITE, ЕЛЕКТРОННИЙ ЖУРНАЛ.

ABSTRACT

The master's thesis contains 138 pages, 17 tables, 37 drawings, a list of used literature of 30 titles, 3 appendices.

The Ukrainian education system is being reorganized and showing good results. However, today, many educational aspects need improvement. This is because students do not have sufficient information about their own performance and their rating, which lowers the motivation for learning that is needed to learn the material better.

Teachers in the modern education system spend a lot of time filling out various paper documents, including journals, rather than devoting more time to the learning process. Current grade in the journal is an important factor in student success. Final / module assessment in educational institutions is considered as a significant factor in predicting the success of knowledge in the framework of the year / semester assessment in the subject. This is why systematic journaling is necessary, but working with paper takes much more time than keeping an electronic journal.

Therefore, the introduction of an electronic journal of accounting of student achievement and attendance will improve the quality of educational services and change the corruption component in education.

Keywords: OBJECT PROGRAMMING LANGUAGE, JAVA, JAVAFX, ECLIPSE, SQLITE, ELECTRONIC JOURNAL.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення.

ПК – персональний комп’ютер.

UML – Unified modelling language, уніфікована мова моделювання, уніфікована мова графічного представлення та об’єктного моделювання в області розробки програмного забезпечення парадигми об’єктно-орієнтованого програмування.

БД – база даних.

ООП – об’єктно-орієнтоване програмування, парадигма програмування, в якій основою є класи та об’єкти, які між собою взаємодіють.

СУБД – система управління базою даних.

СКБД – система контролю базою даних.

IDE – Integrated Development Environment, інтегроване середовище розробки

JVM – віртуальна машина виконання програм, основна частина виконуваної системи Java. Виконує байт-код компільованої програми.

ЗМІСТ

ВСТУП

Актуальність. Українська система освіти реорганізовується та показує досить хороші успіхи. Але, на сьогодні, є багато освітянських аспектів, які потребують покращення. В основному це стосується того, що учні/студенти не володіють достатньою інформацією про власну успішність та свій рейтинг, що знижує мотивацією до навчання, яка необхідна для кращого засвоєння навчального матеріалу.

Викладачі та вчителі в сучасній системі освіти витрачають багато часу на заповнення різноманітних паперових документів, зокрема журналів, замість того, щоб більше часу приділяти процесу навчання. Поточна оцінка в журналі є важливим фактором успішності учня/студента [1]. Підсумкова/ модульна оцінка в освітніх закладах розглядається як істотний фактор прогнозу успішності знань у рамках річної/семестрової оцінки по предмету/дисципліні. [2]. Саме тому систематичне виставлення оцінок у журнали необхідно, але робота із паперовими носіями потребує набагато більше часу ніж ведення електронного журналу.

На сьогоднішній день в Україні є велика кількість закладів освіти, мета яких – якісно та в повній мірі надати учням/студентам освітній рівень, відповідно до рівня навчального закладу. Станом на 2018 р. в Україні нараховується більше 15 тисяч закладів загальної середньої освіти та більше 600 закладів вищої освіти [3].

Україна, як і багато інших розвинених держав використовує модульну систему освіти. Модуль – це логічно завершена система теоретичних знань та фактичних умінь з даної навчальної дисципліни, адаптованих до індивідуальних особливостей суб’єктів учіння з визначеним оптимальним часом на організацію її засвоєння [4]. Розбиття навчального матеріалу на модулі покращує сприйняття нового матеріалу студентами. Введення

модульної системи також робить систему оцінювання знань більш прозорою. Але, через це викладачам необхідно проводити розрахунок успішності та відвідуваності кожного студента в кінці кожного модуля, що може займати багато часу. Впровадження системи державних стандартів освіти також додає навантаження на працівників системи освіти, що підштовхує освітян до пошуку нових методів оптимізації роботи у навчальних закладах.

Впровадження системи державних стандартів освіти також додає навантаження на працівників системи освіти, що підштовхує освітян до пошуку нових методів оптимізації роботи у навчальних закладах.

Мета роботи. Вирішення згаданих проблем системи освіти можна частково знівелювати розробкою та впровадженням електронного журналу обліку успішності та відвідуваності студентів. Використання такого журналу дозволить зекономити багато часу викладачів, адже їм необхідно буде витрачати менше часу на аналіз успішності студентів у кінці модуля. Вчителі зможуть якісніше використовувати цей час для того, щоб краще підготувати навчальний матеріал, ретельніше перевіряти роботи учнів/студентів, шукати індивідуальний підхід до кожного учня/студентів та багато іншого. А це, у свою чергу, значно поліпшить рівень наданих освітянських послуг та підвищить якість знань із предмету/дисципліни [5].

1 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

1.1 Вимоги до структури і функціонування системи

До програми було поставлено перелік вимог:

- мультиплатворменість (використання на комп’ютерах з операційними системами Windows чи MacOS);
- простота інтерфейсу;
- можливість використовувати на різних комп’ютерах.

Також, програма має виконувати свою пряму функцію як електронного журналу – облік відвідування та успішності учнів.

Для цього, вчитель повинен зайти в програму. Програма має надати вчителю список груп на вибір, урок в якій записати в журнал. Список має складатися з груп, заняття в яких проводиться в поточний день (в групі є заняття в поточний день тижня та ще триває період навчання групи). При бажанні, список можна розширити, щоб він складав усі групи і навіть ті, навчання в яких на поточний день не заплановано.

Після вибору бажаної групи, має бути зображено пронумерований список групи. Якщо в програму вже було записано дані за сьогодні, то програма має відобразити ці дані (оцінки та відвідуваність). Після відзначення відвідуваності та виставлення оцінки за урок, вчитель вносить введені дані в програму.

Для того, щоб мати в яку групу вводити, необхідно мати в програмі можливість створити групу. Було б добре й мати можливість її редагувати та видаляти з програми.

Для створення групи, необхідно натиснути на відповідну кнопку, внести назву нової групи та натиснути на кнопку створення. В ході розробки було зроблено рішення, що оптимально буде створювати «порожню групу», яка не матиме днів тижня навчання, яка матиме нелогічний термін навчання (з дня її створення по той ж самий день) та без учнів і, відповідно, записів присутності та успішності.

Надалі потрібна можливість редагувати щойно створену групу (або вже існуючу), змінюючи назву, дні навчання групи та її період навчання. Також необхідно додавати/вилучати учнів з групи. Також бажано зберігати в програмі дані про день народження та, відповідно, вік учнів.

Необхідно також, щоб в програмі була функція перегляду записів журналу певної групи за певний вибраний період. Також необхідно реалізувати функцію редагування запису в групі за минулий день.

Не менш важливою є й можливість перегляду статистики учня. Дана статистика повинна включати в себе середній бал, кількість відсутностей/присутностей за певний вибраний період. Бажано ще й надати графічне представлення цих статистичних даних (графіки).

Для зручності використання програми на різних комп'ютерах, рекомендовано дані, збережені в програмі, зберігати окремо від самої програми, мати змогу в програмі вибирати файл даних, який використовувати. Відповідно, необхідна можливість створення Порожнього файлу даних, з потрібною структурою, але без даних.

Розроблена програма має бути інтуїтивно зрозумілою, щоб користувач міг розібратися з її базовими функціями без попереднього навчання користування програмою чи прочитання інструкції по програмі.

1.1.1 Пошук актантів та варіантів використання

Для створення архітектури системи, визначимо акторів (таблиця 1.1).

Таблиця 1.1 – Актори системи

Актор	Позначення	Дії
Викладач	В	Внесення, редагування та перегляд даних про відвідуваність та успішність студентів.

Присутність в системі лише одного актора сильно полегшує подальше моделювання системи.

Визначимо основні сутності проекту (таблиця 1.2).

Таблиця 1.2 – сутності проекту

Сутність	Позначення	Призначення
Студент (учень)	St	Зберігання даних про день народження
Група (клас)	Gr	Групування учнів по групах
День	D	Зберігання даних про відвідуваність та успішність кожного учня певної групи в певний день
Журнал	J	Зберігання днів занять певної групи

Після проведеного моделювання системи виявлено, що найкраще для подальшого моделювання підходить парадигма програмування ООП.

Об'єктно-орієнтоване програмування (ООП; іноді об'єктозорієнтоване програмування, об'єктоорієнтоване програмування) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція. Одною з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих, коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання

програм. Сьогодні багато мов програмування або підтримують ООП (PHP, Lua) або ж є цілком об'єктно-орієнтованими (зокрема, Java, C#, C++, Python, Ruby і Objective-C, ActionScript 3, Swift, Vala).

Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування Симула в 1960-тих роках, одночасно з посиленням дискусій про кризу програмного забезпечення. Разом із тим, як ускладнювалось апаратне та програмне забезпечення, було дуже важко зберегти якість програм. Об'єктно-орієнтоване програмування частково розв'язує цю проблему шляхом наголошення на модульності програми.

На відміну від традиційних поглядів, коли програму розглядали як набір підпрограм, або як перелік інструкцій комп'ютеру, ООП програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожен об'єкт здатний отримувати повідомлення, обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт — своєрідний незалежний автомат з окремим призначенням та відповідальністю.

1.1.2 Визначення варіантів використання

Так як при виявлення акторів програмної системи був виділений лише один актор, то всі варіанти використання будуть пов’язані тільки з ним. Сформуємо таблицю варіантів використання програмного продукту (таблиця 1.3).

Таблиця 1.3. Варіанти використання

№	Назва	Актор	Короткий опис
В1	Заповнення журналу проведеного уроку	Викладач	Внесення даних про відвідуваності та успішності студентів за щойно проведене заняття
В2	Перегляд журналу за попередні дні	Викладач	Перегляд даних відвідуваності та успішності студентів за попередні заняття
В3	Перегляд статистики учня	Викладач	Перегляд даних відвідуваності та успішності студента. Перегляд графіків успішності та відвідуваності.
В4	Зміна даних журналу за певне попереднє заняття	Викладач	Редагування даних відвідуваності та успішності студентів за певне попереднє заняття з певного предмету
В5	Створення групи	Викладач	Створення в системі групи для навчання
В6	Додавання студентів до групи	Викладач	Додавання до списку студентів групи нового студента
В7	Видалення студентів з групи	Викладач	Видалення студента з списку студентів групи
В8	Редагування даних студента	Викладач	Редагування даних студента (ім’я, дата народження, стать, дані контактної особи, тощо)
В9	Редагування даних групи	Викладач	Редагування даних групи (назва групи, термін навчання, дні тижня, в які відбуваються заняття)

Зобразивши дані ВВ на схемі, отримаємо UML схему варіантів використання (рисунок 1.1).

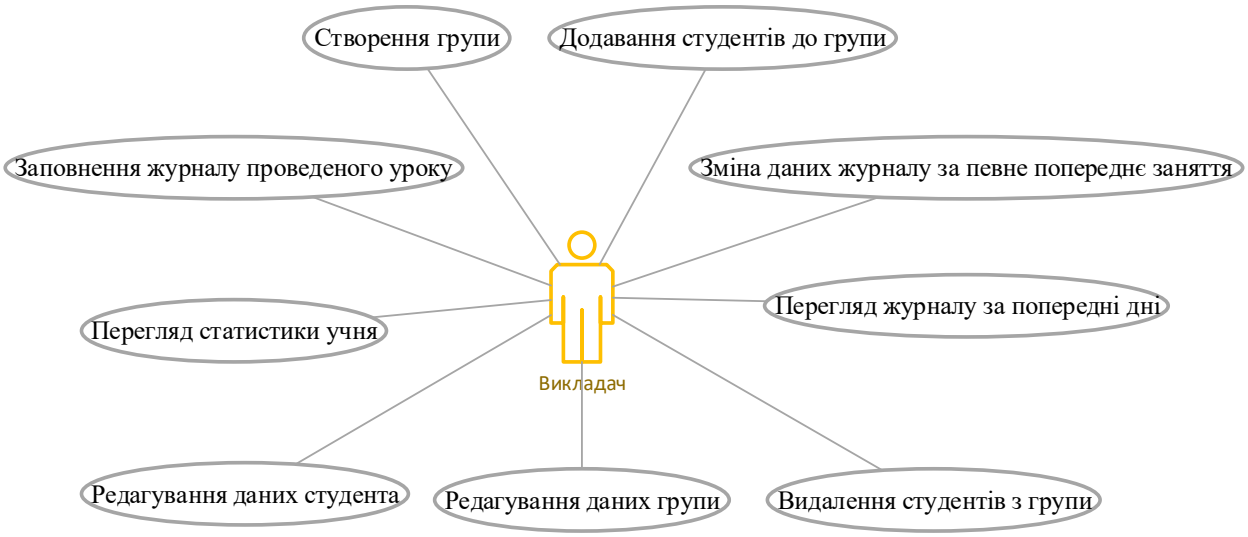


Рисунок 1.1 – Схема варіантів використання

Тепер розглянемо сценарій наведених ВВ ПЗ. Почнімо з ВВ В1 (таблиця 1.4).

Таблиця 1.4. Сценарій ВВ В1 «Заповнення журналу проведеного уроку»

Крок	Дія
1	Вибрати групу, заняття в якій було проведено.
1.1	Заняття в групі було проведено не за графіком занять групи, відзначити прапорець «Не за графіком», і вибрати потрібну групу
2	Внести дані відвідуваності студентів, відмітивши прапорці відвідуваності в таблиці навпроти присутніх студентів
3	Внести дані успішності студентів, ввевши дані в поле оцінки навпроти студента, для якого є оцінка
4	Натиснути кнопку підтвердження внесення даних в програму

Розглянемо сценарій наступного ВВ – В2 «Перегляд журналу за попередні дні» (таблиця 1.5).

Таблиця 1.5 – Сценарій В2 «Перегляд журналу за попередні дні»

Крок	Дія
1	Відкрити журнал необхідної групи в головному вікні
1.1	Якщо в списку немає необхідної групи, відзначити прапорець «Не за графіком», і вибрати потрібну групу
2	У нововідкритому вікні вибрати період, за який необхідно переглянути історію журналу групи
2.1	При потребі перегляду історії журналу ляше для інформації про одного/декількох студентів, а не всіх, необхідні студент/студенти вибираються та натискається кнопка «показати вибрані». Для повернення показу до попереднього стану, необхідно натиснути ту ж кнопку

Розглянемо сценарій наступного ВВ – В3 «Перегляд статистики учня» (таблиця 1.6).

Таблиця 1.6 – Сценарій В3 «Перегляд статистики учня»

Крок	Дія
1	Вибрати необхідного учня в таблиці журналу поточного дня
2	Натискання кнопки «Інформація про учня»
3	Перегляд інформації про учня в нововідкритому вікні

Розглянемо сценарій наступного ВВ – В4 «Зміна даних журналу за певне попереднє заняття» (таблиця 1.7).

Таблиця 1.7 – Сценарій В4 «Зміна даних журналу за певне попереднє заняття»

Крок	Дія
1	Вибір необхідної групи в випадаючому списку головного вікна програми
1.1	Якщо в списку немає необхідної групи, відзначити прапорець «Не за графіком», і вибрати потрібну групу
2	Натиснути кнопку «Повний журнал»
3	В нововідкритому вікні повного журналу вибраної групи, натиснути кнопку «Редагувати»
4	В нововідкритому вікні редагування даних про попередні заняття, в спадаючому списку днів попередніх занять, вибрати необхідну дату
5	В таблиці змінити оцінки та дані про відвідуваність студентів на необхідні
6	Підтвердити зміни, натиснувши клавішу «Редагувати»

Розглянемо сценарій наступного ВВ – В5 «Створення групи» (таблиця 1.8).

Таблиця 1.8 – Сценарій В5 «Створення групи»

Крок	Дія
1	В головному вікні програми перейти в закладку «Управління групами»
2	Натиснути кнопку «Створити групу»
3	В новоствореному вікні ввести назву групи та натиснути кнопку «Ок» для підтвердження створення нової групи з введеним ім’ям

Розглянемо сценарій наступного ВВ – В6 «Додавання студентів до групи» (таблиця 1.9).

Таблиця 1.9 – Сценарій В6 «Додавання студентів до групи»

Крок	Дія
1	В головному вікні програми перейти в закладку «Управління групами»
2	В спадаючому вікні вибору групи вибрати необхідну групу
3	Натиснути кнопку «Додати учня»
4	В новоствореному вікні ввести дані про учня: ім’я, прізвище, стать, дані контактної особи
5	Для підтвердження додавання учня з введеними даними в вибрану групу, натиснути кнопку «Зберегти»

Розглянемо сценарій наступного ВВ – В7 «Видалення студентів з групи» (таблиця 1.10).

Таблиця 1.10 – Сценарій В7 «Видалення студентів з групи»

Крок	Дія
1	В головному вікні програми перейти в закладку «Управління групами»
2	В спадаючому вікні вибору групи вибрати необхідну групу
3	В списку студентів групи вибрати необхідного студента
4	Натиснути кнопку «Видалити студента»
5	В нововідкритому вікні підтвердити свій вибір, натиснувши кнопку «Ок»

Розглянемо сценарій наступного ВВ – В8 «Редагування даних студента» (таблиця 1.11).

Таблиця 1.11 – Сценарій В8 «Редагування даних студента»

Крок	Дія
1	В головному вікні програми перейти в закладку «Управління групами»
2	В спадаючому вікні вибору групи вибрати необхідну групу
3	В списку студентів групи вибрати необхідного студента
4	Натиснути кнопку «Змінити»
5	В новоствореному вікні з даними вибраного студента змінити показані дані на необхідні
6	Підтвердити зміни, натиснувши кнопку «Зберегти»

Розглянемо сценарій наступного ВВ – В9 «Редагування даних групи» (таблиця 1.12).

Таблиця 1.12 – Сценарій В9 «Редагування даних групи»

Крок	Дія
1	В головному вікні програми перейти в закладку «Управління групами»
2	В спадаючому вікні вибору групи вибрати необхідну групу
3	Змінити дані групи (назву, термін навчання та дні, в які проводяться заняття) на необхідні
4	Підтвердити зміни, натиснувши кнопку «Зберегти»

1.2. Проектування програмної системи

1.2.1 Вибір процесу розробки

Почнемо вибір процесу розробки з розгляду найпопулярніших на даний момент.

Гнучка розробка програмного забезпечення (англ. Agile software development, agile-методи) включає в себе різні підходи до розробки

програмного забезпечення, згідно з якими вимоги та рішення розвиваються завдяки спільним зусиллям самоорганізуючих і крос-функціональних команд та їх клієнтів (ів)/кінцевих споживачів. Він виступає за адаптивне планування, еволюційний розвиток, ранні досягнення та постійне вдосконалення , і це заохочує швидку та гнучку реагування на зміни.

Термін agile (іноді пишуться Agile) був популяризований у цьому контексті Маніфестом розвитку спритного програмного забезпечення. Цінності та принципи, викладені в цьому маніфесті, випливають із основи широкого кола програм розробки програмного забезпечення, включаючи Scrum і Kanban.

Хоча є багато анекдотичних доказів того, що прийняття спритних практик та цінностей покращує спритність фахівців з програмного забезпечення, команд та організацій, деякі емпіричні дослідження спростували це свідчення.

Принципи розробки програмного забезпечення Agile

Маніфест гнучкої розробки програмного забезпечення на основі дванадцяти принципів:

- Задоволення клієнтів шляхом ранньої та постійної доставки цінного програмного забезпечення.
- Ласкаво просимо змінити вимоги навіть у пізньому розвитку.
- Поставляйте робоче програмне забезпечення часто (тижні, а не місяці)
- Тісна, щоденна співпраця між діловими людьми та розробниками
- Проекти будуються навколо мотивованих людей, яким слід довіряти
- Розмова віч-на-віч - найкраща форма спілкування (спільне розташування)
- Робоче програмне забезпечення є основним показником прогресу
- Сталий розвиток, здатний підтримувати постійний темп

– Постійна увага до технічної досконалості та гарного дизайну
– Простота - мистецтво максимізувати обсяг незайнятої роботи - важливо
– Найкращі архітектури, вимоги та проекти випливають із самоорганізуючих команд
– Регулярно команда замислюється над тим, як стати ефективнішою, і відповідно корегується (рисунок 1.2). [6]

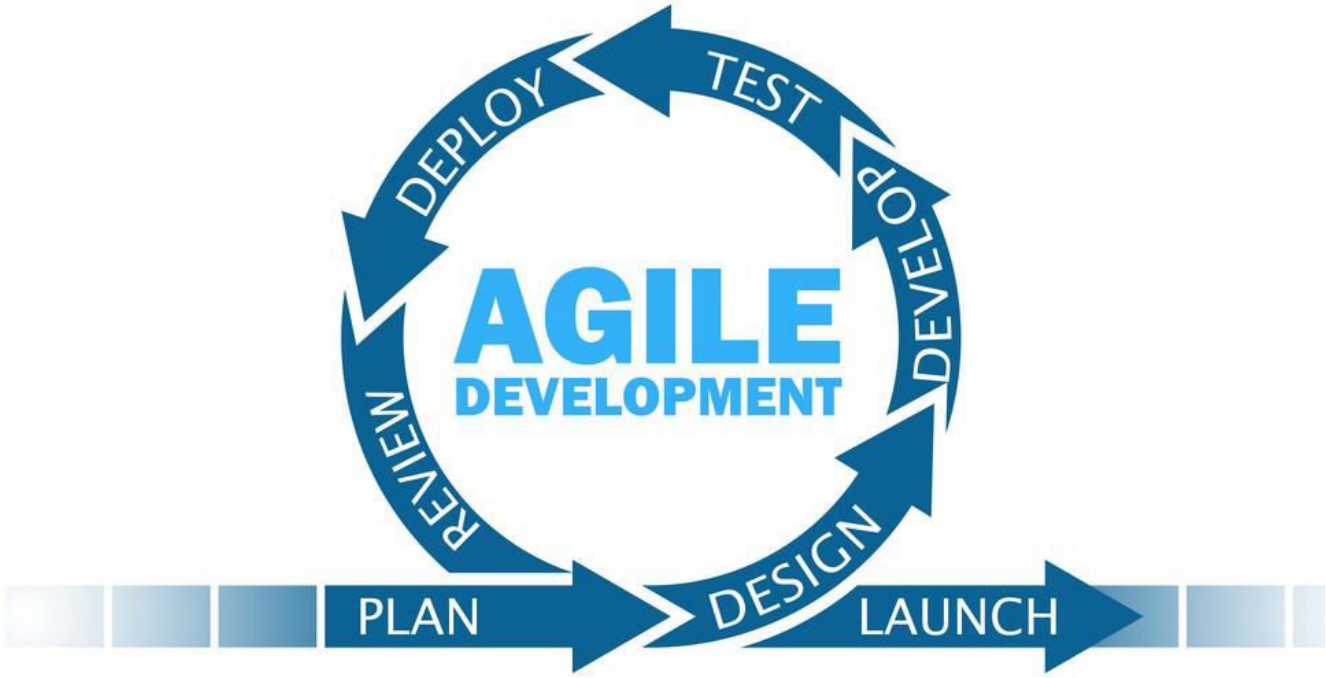


Рисунок 1.2 – Agile

Водоспад модель була першою моделлю процесу повинні бути введений. Це дуже просто зрозуміти і використовувати. У моделі водоспаду кожна фаза повинна бути завершена до того, як наступна фаза може розпочатися, і фаз не буде перекриватися. Модель водоспаду - це самий

ранній підхід розробки, який застосовувався для розробки програмного забезпечення.

У підході «Водоспад» весь процес розробки програмного забезпечення розділений на окремі фази. Результат однієї фази виконує функцію входу для наступної фази послідовно. Це означає, що будь-яка фаза процесу розвитку починається лише за умови завершення попередньої фази. Модель водоспаду – це послідовний процес проектування, в якому прогрес сприймається як постійно протікає вниз (як водоспад) через фази зачаття, ініціації, аналізу, проектування, будівництва, тестування, виробництва/впровадження та обслуговування.

Як модель водоспаду ілюструє процес розробки програмного забезпечення в лінійному послідовному потоці; отже, її також називають лінійно-послідовною моделлю життєвого циклу .

Послідовні фази в моделі водоспаду:

- Вимоги: Перший етап включає розуміння того, що потрібно розробити та яка його функція, призначення тощо. Тут специфікації вводу та виходу або кінцевого продукту вивчаються та маркуються.
- Проектування системи: На цій фазі вивчаються технічні вимоги з першого етапу і готується проектування системи. Дизайн системи допомагає у визначенні апаратних та системних вимог, а також допомагає визначити загальну архітектуру системи. Програмний код, який слід записати на наступному етапі, створюється зараз.
- Реалізація: За допомогою входів від проектування системи система спочатку розробляється в невеликих програмах, званих одиницями, які інтегруються в наступний етап. Кожен пристрій розробляється та перевіряється на свою функціональність, яку називають модульним тестуванням.
- Інтеграція та тестування: Усі підрозділи, розроблені на етапі впровадження, інтегруються в систему після тестування кожного

блоку. Розроблене програмне забезпечення повинно пройти постійне тестування програмного забезпечення, щоб з'ясувати, чи є якісь недоліки чи помилки. Тестування проводиться так, щоб клієнт не стикався з жодними проблемами під час встановлення програмного забезпечення.

- Розгортання системи: Після того, як буде проведено функціональне та нефункціональне тестування, виріб буде розгорнуто в середовищі споживачів або випущено на ринок.
- Технічне обслуговування: Цей етап відбувається після встановлення та включає в себе зміни в системі або окремому компоненті для зміни атрибутів або підвищення продуктивності. Ці модифікації виникають або через запити на зміни, ініційовані замовником, або дефекти, виявлені під час використання живої системи. Клієнту надається регулярне обслуговування та підтримка розробленого програмного забезпечення.

Всі ці фази є каскадними одна до одної, коли прогрес сприймається як постійно протікає вниз (як водоспад) крізь фази. Наступний етап починається лише після того, як визначений набір цілей буде досягнуто для попереднього етапу і він підписаний, тому назва " Модель водоспаду ".

Переваги моделі водоспаду:

- Перевага розвитку водоспаду полягає в тому, що він дозволяє здійснювати департаменталізацію та контроль. Графік може бути встановлений з термінами для кожного етапу розробки, і продукт може проходити по етапах моделі розробки по черзі.
- Модель водоспаду прогресує через легко зрозумілу і пояснювану фази, і тому вона проста у використанні.
- Керувати нею легко через жорсткість моделі - кожна фаза має конкретні результати та процес перегляду.

– У цій моделі фази обробляються та завершуються одна за одною і вони не перетинаються. Модель водоспаду добре працює для менших проектів, де вимоги дуже добре вивчені.

Недоліки моделі водоспаду:

– Важко оцінити час та витрати на кожну фазу процесу розробки.
– Після того, як заявка знаходиться на етапі тестування, дуже важко повернутися назад і змінити щось, що не було продумано на етапі концепції.
– Недобра модель для складних та об'єктно-орієнтованих проектів.
– Не підходить для проектів, де вимоги мають помірний та високий ризик зміни. (рисунок 1.3). [7]

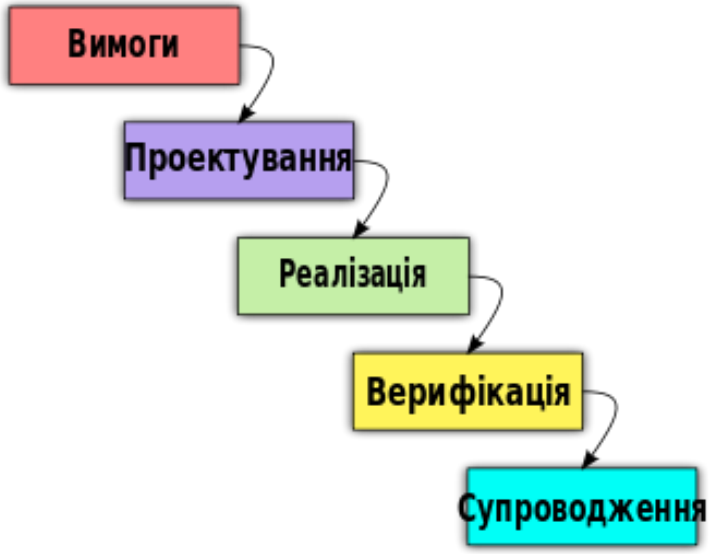


Рисунок 1.3 – Водоспадна модель

Екстремальне програмування – один із кількох популярних процесів Agile . Перший проект екстремального програмування розпочався 6 березня 1996 року. В багатьох компаніях різного розміру та галузей у всьому світі це було дуже успішно.

Екстремальне програмування є успішним, оскільки підкреслює задоволеність клієнтів. Замість того, щоб доставити все, що ви, можливо,

захочете в якусь дату в майбутньому, цей процес доставляє програмне забезпечення, яке вам потрібно, як вам потрібно. Екстремальне програмування дає змогу вашим розробникам впевнено реагувати на зміни вимог замовника, навіть у кінці життєвого циклу.

Екстремальне програмування наголошує на роботі в команді. Менеджери, замовники та розробники - всі рівноправні партнери в колективі. Екстремальне програмування реалізує просте, але ефективне середовище, що дозволяє командам стати високопродуктивними. Команда самоорганізовується навколо проблеми, щоб вирішити її якомога ефективніше.

Екстремальне програмування покращує програмний проект п'ятьма важливими способами; спілкування, простота, відгуки, повага та мужність. Екстремальні програмісти постійно спілкуються зі своїми клієнтами та колегами-програмістами. Вони зберігають свою конструкцію простою та чистою. Вони отримують зворотній зв'язок, тестуючи своє програмне забезпечення, починаючи з першого дня. Вони доставляють систему клієнтам якомога раніше та впроваджують зміни, як було запропоновано. Кожен невеликий успіх поглиблює їхню повагу до унікального внеску кожного члена команди. За допомогою цієї програми Екстремальні програмісти можуть сміливо реагувати на мінливі вимоги та технології.

Найдивніший аспект Екстремального програмування - це його прості правила. Екстремальне програмування дуже схоже на пазл. Є багато маленьких шматочків. Окремо шматки не мають сенсу, але в поєднанні разом можна побачити повну картину. Правила можуть здаватися спочатку незручними та, можливо, навіть наївними, але базуються на здорових цінностях та принципах.

Наші правила встановлюють очікування між членами команди, але самі по собі не є кінцевою метою. Ви зрозумієте, що ці правила визначають середовище, що сприяє співпраці в команді та розширенню можливостей, це ваша мета. Після досягнення продуктивної роботи в команді буде

продовжуватися навіть тоді, коли правила будуть змінені відповідно до конкретних потреб вашої компанії.

Клієнти полюбляють бути партнерами в процесі програмного забезпечення, розробники активно роблять внесок незалежно від рівня досвіду, а менеджери зосереджуються на спілкуванні та відносинах. Непродуктивна діяльність була урізана, щоб зменшити витрати і розчарування всіх учасників (рисунок 1.4). [8]

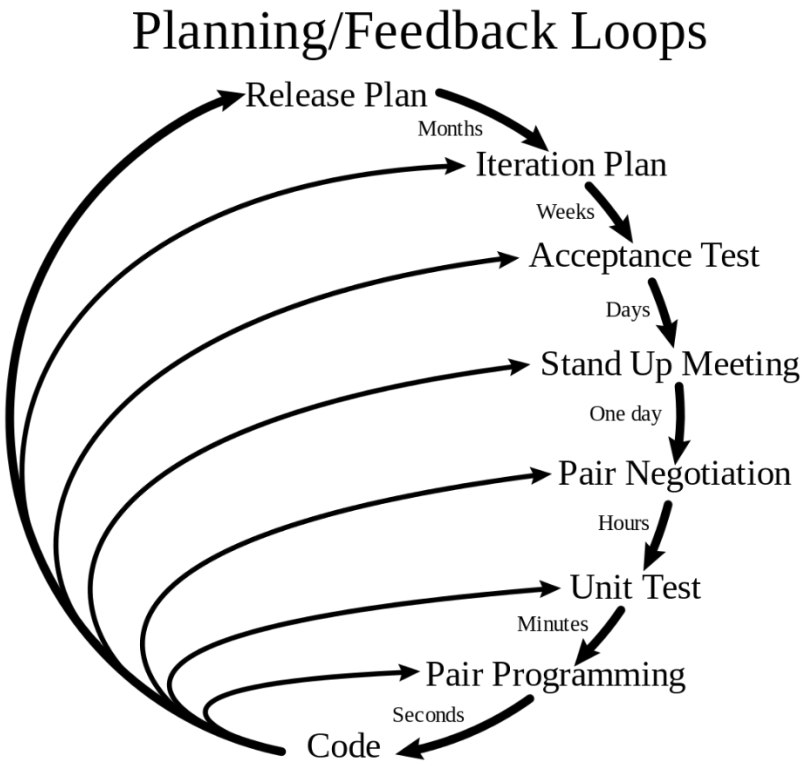


Рисунок 1.4 – Екстремальне програмування

Scrum – тип Agile Framework . Це фреймворк, в рамках якого люди можуть вирішити складну адаптивну проблему, тоді як продуктивність та креативність представлення продукту – це найважливіші показники. Scrum використовує ітеративний процес.

Головними рисами Scrum є:

– Scrum - це легковаговима рамка

– Scrum підкреслює самоорганізацію

– Scrum зрозумілий просто
– Scrum Framework допомагають команді працювати разом

Життєвий цикл Scrum:

– Спринт – це тайм-тайм, який становить один місяць або менше. Новий спринт починається одразу після завершення попереднього спринту.
– Випуск: після закінчення продукту він переходить до стадії випуску.
– Огляд спринту: якщо продукт все ще має деякі недосяжні функції, він буде перевірений на цій стадії, після чого продукт буде переданий на ретроспективну стадію спринту.
– Ретроспектива спринту: на цьому етапі перевіряється якість або стан продукту.
– Блокування продукту: відповідно до особливостей пріоритетності, продукт організований.
– Блокування спринту: блокування спринту розділено на дві частини, призначені для продукту функціями спринту та наради планування спринту.

Перевага використання рам Scrum:

– Фреймворк Scrum – це швидко рухається та ефективно працює з грошима.
– Фреймворк Scrum працює, поділяючи великий продукт на невеликі субпродукти. Це як стратегія поділу та перемоги
– У Scrum задоволення клієнтів дуже важливе.
– Scrum має адаптивний характер, оскільки має короткий спринт.
– Оскільки Фреймворк Scrum покладаються на постійний зворотний зв'язок, тому якість продукції зростає за менший проміжок часу

Недоліки використання рамки Scrum:

– Scrum Framework не дозволяють змінити свій спринт.

– Scrum Framework не повністю описана модель. Якщо ви хочете його прийняти, вам потрібно заповнити рамку своїми власними деталями, такими як Extreme Programming (XP), Kanban, DSDM.
– Scrum може бути важким для планування, структуризації та організації проекту, який не має чіткого визначення.
  – Щоденні зустрічі Scrum та часті огляди потребують значних ресурсів. (рисунок 1.5). [9]

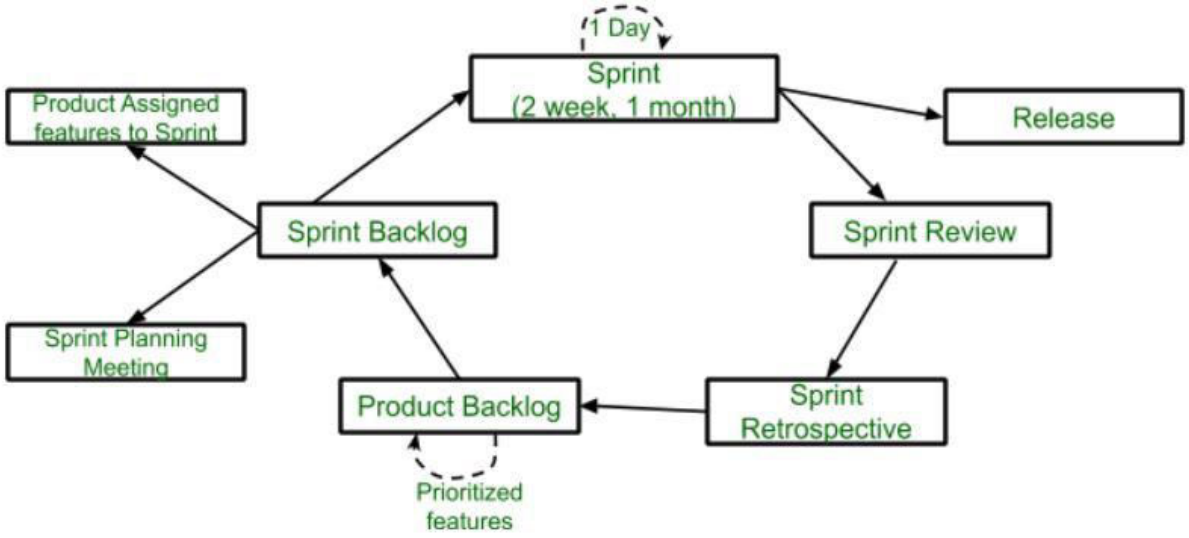


Рисунок 1.5 – Scrum

Після аналізу вище перерахованих методик розробки програмного забезпечення (ПЗ), було обрано Agile, так як дана методика найкраще підходить на цей проект розробки ПЗ. Agile легко використовувати якраз на розробку невеликих проектів з малою командою та можливістю постійно мати змогу протестувати продукт на перевірку на несправності (баги) та відповідність вимогам.

1.2.2 Побудова схеми бази даних

Як базу даних було обрано SQLite через її легку портабельність та простоту розгортання на комп’ютерах. Для початку, слід визначити схему бази даних (БД). (Рисунок 1.6)

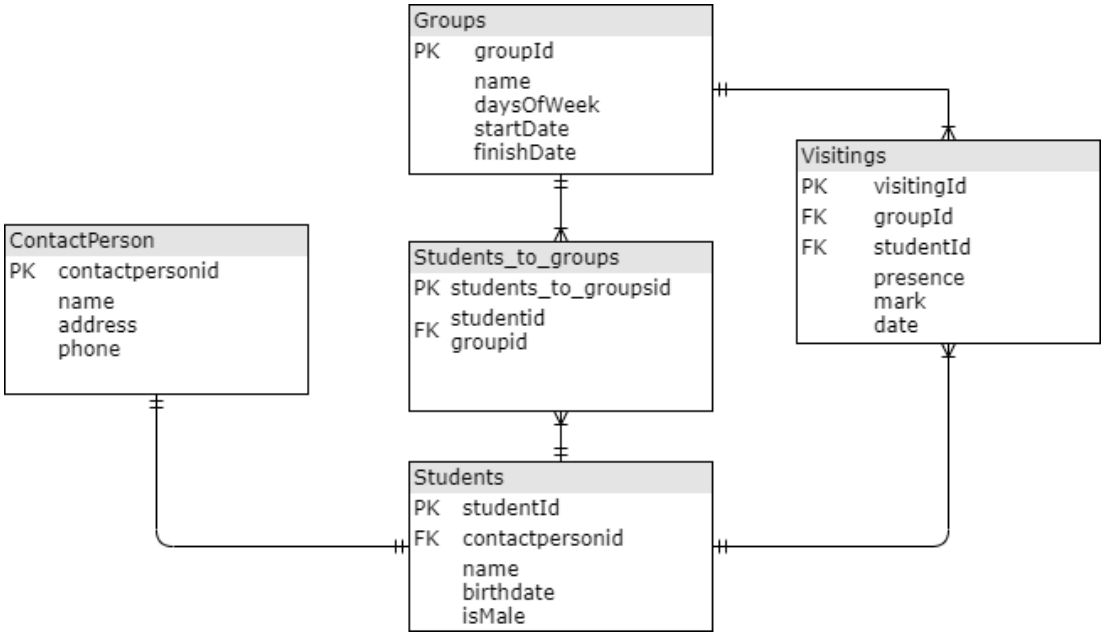


Рисунок 1.6 – Схема бази даних

В таблиці Groups зберігаються сутності груп. В даній таблиці також зберігаються дані про ім’я групи, дні роботи групи, дати початку та кінця навчального процесу групи.

В таблиці Students зберігаються сутності учнів. В даній таблиці також зберігаються дані про ім’я учня , його день народження та стать. Також є зовнішній ключ з таблицею Groups для зв’язку сутностей учнів та групи. Тобто, кожен учень в таблиці прив’язаний до певної групи.

В таблиці Visitings зберігаються сутності відвідувань. В даній таблиці зберігаються дані про дату відвідування, оцінку та присутність. Також є два зовнішні ключі: для зв’язку з певною сутністю учня та групи. Тобто, кожен запис відвідувань зв’язаний з одною сутністю учня та одною сутністю групи.

Для попереднього створення, редагування, менеджменту та управління файлом БД, необхідна система управління базою даних (СУБД). Для SQLite є

досить велика кількість різноманітних СУБД. Було обрано програму «SQLiteStudio» як одну з найпопулярніших та найпростіших програм свого класу для вибраної бази даних.

1.2.3 Моделювання архітектури системи

Як архітектурний шаблон було обрано шаблон MVC.

Шаблон проектування MVC передбачає поділ даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель , уявлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно. Модель (Model) надає дані предметної області поданням і реагує на команди контролера , змінюючи свій стан. Представлення (View) відповідає за відображення даних предметної області (моделі, Model) користувачеві, реагуючи на зміни моделі. контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін.

Шаблон проектування Model-View-Controller (далі просто MVC) ліг в основу архітектурного рішення першого середовища програмування з графічним інтерфейсом користувача – Smalltalk-80 . Вперше MVC описав ще в 1978 році норвежець Трюгве Рінскауг, який працював деякий час в лабораторії Xerox PARC. Реалізацію шаблону в Smalltalk-80 описав трохи пізніше Стів Бурбек (рисунок 1.7).

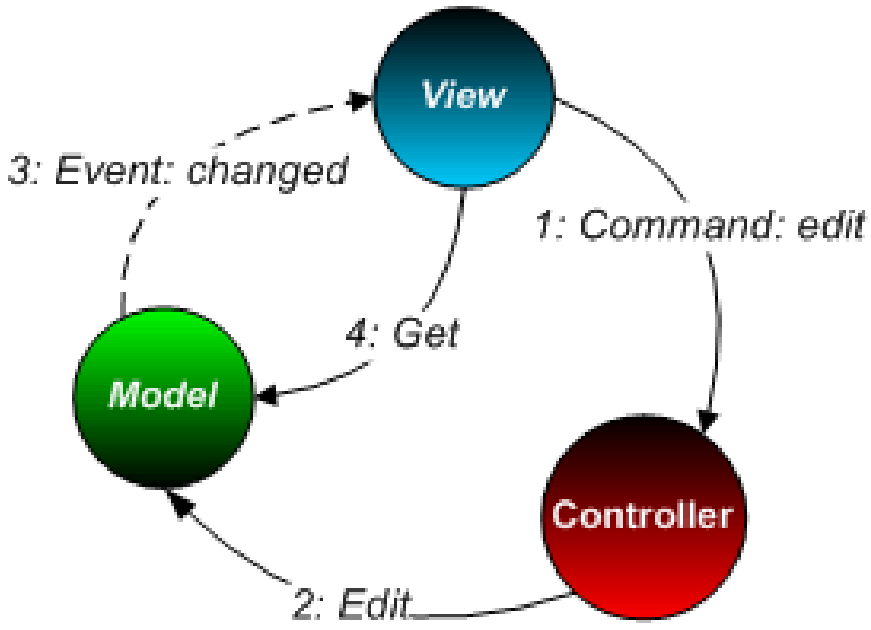


Рисунок 1.7 – MVC

Розглянемо простий приклад MVC, який згодом буде реалізований в презентаційних програмах, які додаються до статті. Модель може являти собою об'єкт, який реалізує перемикач. У найпростішому випадку дана модель характеризується станом: вимкнений або включений - і, крім того, дозволяє змінювати його - об'єкт моделі має метод зміни стану: вимкнути і включити. Подання відображає на дисплеї стан перемикача за допомогою певної текстової або графічної форми. Наприклад, уявлення може відображати текстову мітку, яка при зміні стану моделі перемикача відобразить відповідний текст: «вимкнений» або «включений». Крім відображенняуявлення дозволяє користувачеві змінювати стан перемикача за допомогою графічних примітивів, наприклад, двох кнопок з написами: «Включити» і «Вимкнути». Подання вміє тільки відображати стан моделі перемикача, для зміни уявлення звертається до контролера . Контролер являє собою об'єкт, який в нашому випадку вміє тільки змінювати стан моделі перемикача (рисунок 1.8). [10]

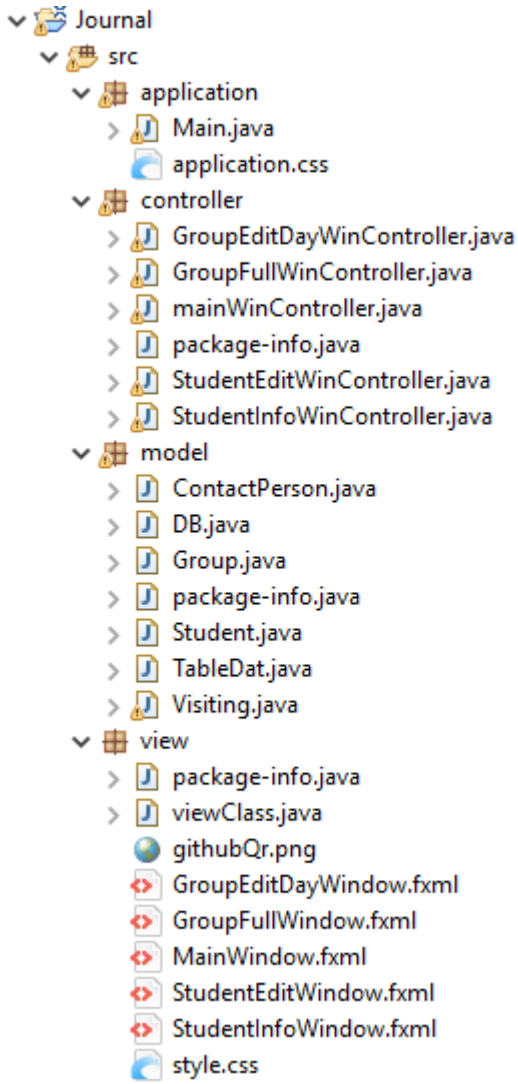


Рисунок 1.8 – Схема класів проекту програми

В пакеті application знаходиться клас Main, з якого і відбувається запуск програми.

В пакеті model знаходяться класи моделі. Тобто, по класу для кожної сутності з бази даних і клас зв’язку з самою БД.

В пакеті view знаходяться файли представлення. Це моделі вікон програми (файли .fxml),файл стилю програми (style.css) та зображення, яке використовується в програмі (QR-код проекту).

В пакеті controller знаходяться класи контролерів. Для кожної моделі вікон програми в відповідність створено клас контролера.

Кожен з останніх трьох пакетів також містить файл package-info, де записана загальна інформація свого пакету.

Вибрана структура проекту програми дозволяє з легкістю будувати зв’язки в програмі, нарощувати програму, додаючи нову логіку чи нове вікно, легко продовжувати оновлення програми, розширюючи її функціонал та швидше знаходити критичне місце в програмі при виявленні несправності чи багу.

1.3. Конструювання програмної системи

1.3.1 Вибір мови та середовища розробки

Після проведеного моделювання системи, можна почати вибір засобів розробки програмної системи.

Почнемо з вибору мови програмування. Розгляньмо найпопулярніші мови програмування. Розглядатимемо ООП мови програмування, так як дана парадигма програмування була обрана під час моделювання.

Java – це комп'ютерна мова програмування загального призначення, яка є одночасною , класно-орієнтованою, об'єктно-орієнтованою та спеціально розробленою, щоб мати якомога менше залежностей від реалізації. Він покликаний дозволити розробникам додатків «писати один раз, запускати куди завгодно» (WORA) , тобто компільований код Java може працювати на всіх платформах, які підтримують Java без необхідності перекомпіляції.

Наприклад, ви можете написати та компілювати програму Java на UNIX та запустити її на машині Microsoft Windows, Macintosh або UNIX без будь-яких змін вихідного коду. WORAдосягається компіляцією програми Java на проміжну мову, яку називають байт-кодом . Формат байт-коду не залежить від платформи . Віртуальна машина, яка називається Java Virtual

Machine (JVM) , використовується для запуску байтового коду на кожній платформі.

Java спочатку була розроблена Джеймсом Гослінгом у компанії Sun Microsystems (яка з тих пір була придбана корпорацією Oracle) і випущена в 1995 році як основний компонент платформи Java Sun Microsystems. Мова отримує значну частину свого синтаксису від C та C++, але має менше можливостей низького рівня, ніж будь-яка з них.

Корпорація Oracle є поточним власником офіційної реалізації платформи Java SE після придбання Sun Microsystems 27 січня 2010 року. Ця реалізація заснована на оригінальній реалізації Java від Sun. Реалізація Oracle доступна для Microsoft Windows, Mac OS X, Linux та Solaris.

Реалізація Oracle складається з двох різних дистрибутивів:

- Середовище виконання Java (JRE), що містить частини платформи Java SE, необхідні для запуску програм Java і призначені для кінцевих користувачів.
- Java Development Kit (JDK) , призначений для розробників програмного забезпечення та включає засоби розробки, такі як компілятор Java, Javadoc, Jar та відладчик.

Java використовує автоматичний збирач сміття для управління пам'яттю в життєвому циклі об'єкта. Програміст визначає, коли створюються об'єкти, а час виконання Java відповідає за відновлення пам'яті, коли об'єкти більше не використовуються. Як тільки жодні посилання на об’єкт не залишаються, недосяжна пам'ять стає правом автоматично звільнятися сміттєзбірником.

Щось подібне до витоку пам'яті може все-таки статися, якщо код програміста містить посилання на об'єкт, який більше не потрібен, як правило, коли об'єкти, які більше не потрібні, зберігаються в контейнерах, які все ще використовуються. Якщо викликаються методи для неіснуючого об'єкта, кидається «NullPointerException».

Збір сміття може відбутися в будь-який час. В ідеалі це відбудеться, коли програма не працює. Він гарантовано спрацьовує, якщо на купі недостатньо вільної пам’яті для виділення нового об’єкта; це може спричинити миттєву зупинку програми. Явне управління пам’яттю неможливо на Java (рисунок 1.9). [11]

Рисунок 1.9 – Java

C # – це сучасна об'єктно-орієнтована мова програмування, розроблена в 2000 році Андерсом Хейлсбергом в Microsoft як суперник Java (на що вона досить схожа). Він був створений тому, що Sun (куплений пізніше Oracle) не хотів, щоб Microsoft вносила зміни в Java , тому Microsoft вирішила створити свою мову замість цього. C# швидко зростав з моменту його створення, завдяки широкій підтримці корпорації Майкрософт, яка допомогла йому отримати велику кількість наступних; зараз це одна з найпопулярніших мов програмування у світі.

C# – це мова загального призначення, розроблена для розробки програм на платформі Microsoft і вимагає, щоб система .NET в Windows працювала. C# часто вважають гібридом, який використовує найкращі C та

C++, щоб створити справді модернізовану мову. Хоча фреймворк .NET підтримує декілька інших мов кодування, C# швидко став однією з найпопулярніших.

C# можна використовувати для створення майже будь-чого, але особливо сильний у створенні настільних додатків та ігор Windows. C# також може використовуватися для розробки веб-додатків і стає все більш популярною і для мобільних розробок . Інструменти для різних платформ, такі як Xamarin, дозволяють додаткам, написаним на C#, користуватися майже на будь-якому мобільному пристрої.

C# широко використовується для створення ігор за допомогою ігрового двигуна Unity , який є найпопулярнішим ігровим двигуном сьогодні. Більше третини найкращих ігор створено з Unity, і є приблизно 770 мільйонів активних користувачів ігор, створених за допомогою двигуна Unity. Unity також використовується для VR, 90% всіх Samsung Gear і 53% всіх ігор Oculus Rift VR розроблені за допомогою Unity.

C# – дуже популярний інструмент для створення цих додатків, і тому робить чудовий вибір для будь-якого програміста, який сподівається проникнути в індустрію розвитку ігор, або для всіх, хто цікавиться віртуальною реальністю.

C# має багато функцій, які полегшують навчання. Це мова високого рівня, відносно проста для читання, при цьому багато найскладніших завдань абстрагуються, тому програмісту не потрібно переживати про них. Наприклад, управління пам'яттю знімається з відповідальності користувача та обробляється схемою збору сміття .NET.

Це також статично набрана мова, тому код перевіряється перед тим, як його перетворити на додаток. Це полегшує пошук помилок, що може бути особливо корисним для новачків.

Хоча синтаксис C# є більш послідовним і логічним, ніж C++, ще є чому навчитися. C# є складною мовою, і оволодіння нею може зайняти більше часу, ніж простіші мови, такі як Python. Це означає, що користувачам

потрібно вивчити значну кількість коду для створення розширених програм, що може бути непридатним для деяких нових користувачів.

Будучи потужним, гнучким і добре підтримуваним, C# швидко став однією з найпопулярніших доступних мов програмування. Сьогодні це четверта найпопулярніша мова програмування, приблизно 31% усіх розробників регулярно використовують її. Це також третя найбільша спільнота в StackOverflow (яка була побудована за допомогою C#) з більш ніж 1,1 мільйонами тем.

Ця популярність перетворюється на процвітаючий ринок праці – щомісяця (у всьому світі) рекламується понад 17 000 робочих місць на C# із середньою зарплатою понад 72 000 доларів . Якщо звужуватися лише до США, щомісяця розміщується понад 6 000 робочих місць із щорічною зарплатою в 92 000 доларів (рисунок 1.10). [12]

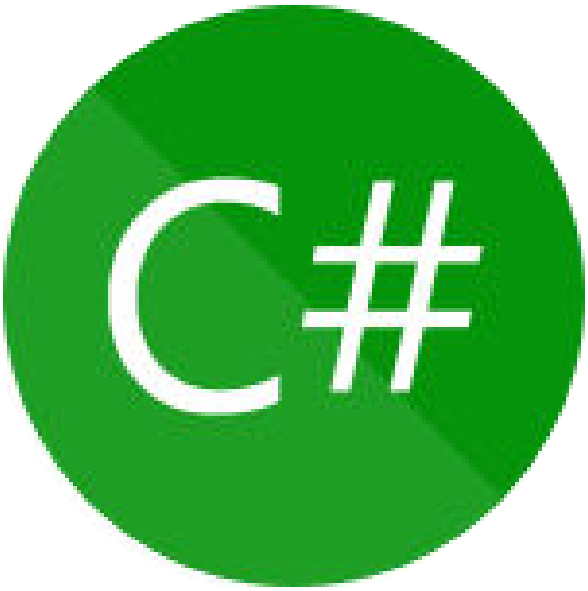


Рисунок 1.10 – C#

C++ – мова програмування, яка має імперативні та об’єктно-орієнтовані функції. Його також називають мовою програмування середнього рівня. Він розроблений Bjarne Stroustrup в лабораторіях Bell з 1979 року. Вперше він з'явився в 1985 році. Він складений, загального призначення, статичного типу, чутливого до регістру та вільної мови програмування. Він

підтримує процедурне, об'єктно-орієнтоване та загальне програмування. Це наявність багатої стандартної бібліотеки з багатим набором функцій, що управляють файлами та методами, що маніпулюють структурами даних тощо.

C++ широко використовується серед програмістів або розробників, головним чином, в області додатків. Вона містить важливі деталі, включаючи ядро мови, надаючи весь необхідний будівельний матеріал, включаючи змінні, тип даних, літерали і т.д. Він підтримує об'єктно-орієнтоване програмування, включаючи його функцію, такі як успадкування, поліморфізм, інкапсуляцію і абстракцію. Ці поняття роблять мову C++ різною і в основному використовується для розробки програм легко та концептуально.

Існує кілька переваг використання C++ для розробки додатків, а багато додатків на основі продукту, розроблених цією мовою, лише завдяки його особливостям та безпеці. Будь ласка, знайдіть нижче розділи, де широко використовувались C++. Нижче наводиться список 10 найкращих застосувань C++.

Застосування: використовується для розробки нових додатків C++. Програми, засновані на графічному інтерфейсі користувача, які дуже використовуються додатки, такі як Adobe Photoshop та інші. Багато додатків систем Adobe розроблені в C++, як Illustrator, прем’єра Adobe Adobe та готові зображення, а розробники Adobe вважаються активними у спільноті C++.

Ігри: Ця мова також використовується для розробки ігор. Він перекриває складність 3D-ігор. Це допомагає оптимізувати ресурси. Він підтримує багатокористувацький варіант з мережею. використання C++ дозволяє процедурне програмування для інтенсивних функцій процесора та забезпечення контролю над обладнанням, і ця мова дуже швидка, через що вона широко використовується при розробці різних ігор або в ігрових двигунах. C++ в основному використовується при розробці наборів ігрового інструменту.

Анімація: Є анімовані програми, розроблені за допомогою мови C++. Програмне забезпечення для 3D-анімації, моделювання, моделювання, візуалізації називають потужним набором інструментів. Він широко застосовується для побудови в режимі реального часу, обробки зображень, мобільних сенсорних додатків та візуальних ефектів, моделювання яких в основному кодується на C++. Це розроблене програмне забезпечення, яке використовується для анімації, оточення, графіки руху, віртуальної реальності та створення персонажів. Віртуальні реальні пристрої є найпопулярнішими в сучасному світі розваг.

Веб-браузер: Ця мова використовується і для розробки браузера. C++ використовується для створення Google Chrome та Інтернет-браузера Mozilla Firefox. Деякі програми написані на C++, звідки браузер Chrome є одним із них, а інші – як файлова система, карта зменшує обробку даних великих кластерів. У Mozilla є інша програма, також написана на C++, що є клієнтом електронної пошти Mozilla Thunderbird. C++ також є механізмом візуалізації проектів Google та Mozilla з відкритим кодом.

Доступ до бази даних: Ця мова також використовується для розробки програмного забезпечення для баз даних або програмного забезпечення з відкритим кодом. Прикладом є MySQL, яка є одним з найбільш популярного управління базами даних програмного забезпечення і широко використовуються в організаціях або серед розробників. Це допомагає заощадити час, гроші, бізнес-системи та пакується програмне забезпечення. Існують і інші додатки, що базуються на доступі до програмного забезпечення баз даних, такі як Wikipedia, Yahoo, youtube тощо. Іншим прикладом є RDBMS Bloomberg, який допомагає інвесторам надавати фінансову інформацію в реальному часі. В основному він написаний на C++, завдяки чому доступ до бази даних є швидким і швидким або точним, щоб доставляти інформацію про бізнес та фінанси, новини по всьому світу.

Доступ до медіа: C++ також використовується для створення медіаплеєра, управління відеофайлами та аудіофайлами. Приклад - програвач

Winamp Media, розроблений мовою C++, що дозволяє нам насолоджуватися музикою, отримувати доступ та ділитися відео та музичними файлами. Він також має такі функції, як підтримка мистецтва, потокове передавання аудіо та відео. Він також забезпечує доступ до Інтернет-радіостанцій.

Компілятори: Більшість компіляторів написані лише мовою C++. Компілятори, які використовуються для компіляції інших мов, таких як C #, Java тощо, переважно написані лише на C++. Він також використовується при розробці цих мов, а також C++ є незалежним від платформи та здатним створювати різноманітне програмне забезпечення.

Операційні системи: Він також використовується для розробки більшості операційних систем для Microsoft та кількох частин операційної системи Apple. Microsoft Windows 95, 98, 2000, XP, office, Internet Explorer та візуальна студія, мобільні операційні системи Symbian в основному написані лише мовою C++.

Сканування: Такі програми, як сканер фільму або сканер камери, також розроблені мовою C++. Він використовувався для розробки технології PDF для друку документації, обміну документом, архівації документа та публікації документів.

Інші сфери використання: використовується для медичних та інженерних застосувань, систем автоматизованого проектування. Ці програми схожі на машини МРТ-сканування, системи CAM, які в основному використовуються в лікарнях, місцевому, державному та національному уряді та інших відділеннях для будівництва та видобутку тощо. Програми C++ вважаються першою бажаною мовою, яку використовують розробники під час роботи вважається для будь-якого розвиваючого додатка.

C++ – це мова, яка використовується скрізь, але в основному для систем програмування та вбудованих систем. Тут системне програмування означає засоби розробки операційних систем або драйверів, які взаємодіють із апаратним забезпеченням. Вбудована система означає речі, які представляють собою автомобілі, робототехніку та техніку. C++ має більш

високу чи багату спільноту та розробників, що допомагає легко набирати розробників та онлайн-рішення легко.

C++ називається найбезпечнішою мовою через його безпеку та особливості. Це перша мова для будь-якого розробника, який зацікавлений працювати в мовах програмування. Це легко вивчити, оскільки це чиста концептуальна мова. Його синтаксис дуже простий, що дозволяє легко писати чи розвивати, а помилки можна легко повторити. Перш ніж використовувати будь-яку іншу мову, програмісти вважали за краще спочатку вивчати C ++, а потім вони використовували інші мови. Але більшість розробників намагаються дотримуватися C++ лише через широке різноманіття використання та сумісність з декількома платформами та програмним забезпеченням (рисунок 1.11). [13]

Рисунок 1.11 – C++

Після аналізу наведених вище мов програмування, було обрано мову програмування Java через такі її переваги:

– мультиплатформеність, що є вимогою до ПЗ;
– велика кількість бібліотек, що дозволить з легкістю розширювати функціонал додатку;

– велика популярність, що дозволить швидко і легко вирішити проблеми під час написання коду в більшості випадків.

Вибравши мову програмування, виберемо середовище розробки.

Eclipse, давно найпопулярніший Java IDE, є безкоштовним та відкритим кодом та пишеться здебільшого на Java, хоча архітектура плагінів дозволяє розширювати Eclipse іншими мовами. Eclipse виник у 2001 році як проект IBM по заміні сімейства IDE, що базується на Smalltalk, IBM Visual Age ID на портативний IDE на базі Java. Метою проекту було затемнення Microsoft Visual Studio, звідси і назва.

Переносимість Java допомагає Eclipse бути кросплатформенним: Eclipse працює на Linux, Mac OS X, Solaris та Windows. Інструментарій інструментів Java Standard Widget (SWT) принаймні частково відповідає за зовнішній вигляд Eclipse, за добрий чи поганий стан. Так само Eclipse завдячує своєю ефективністю (або, як кажуть деякі, її відсутністю) JVM. Eclipse має репутацію роботи повільно, що повертається до старих апаратних засобів та старих JVM. Однак і сьогодні він може відчувати себе повільно, особливо, коли він оновлюється у фоновому режимі, встановивши багато плагінів.

Частина накладних витрат у Eclipse – це вбудований поступовий компілятор, який запускається кожного разу, коли він завантажує файл і щоразу, коли ви оновлюєте свій код. Це в балансі дуже хороша річ, і дає показники помилок під час введення.

Незалежно від системи збирання, проект Eclipse Java також підтримує модель його вмісту, яка включає інформацію про ієрархію типів, посилання та декларації елементів Java. Це також добре врівноважує і дозволяє декілька помічників із редагування та навігації, а також вигляд контуру.

Екосистема плагінів – одна з сильних сторін Eclipse, а також є джерелом випадкових розладів. На даний момент ринок Eclipse містить понад 1600 рішень, а плагіни, що надаються спільнотою, можуть або не

можуть працювати як рекламовані. Проте плагіни Eclipse містять підтримку понад 100 мов програмування та майже 200 рамок розробки додатків.

Більшість серверів Java також підтримується: якщо ви визначите нове підключення до сервера від Eclipse, ви перейдете до списку папок постачальників, під яким ви знайдете близько 30 серверів додатків, включаючи дев'ять версій Apache Tomcat. Комерційні постачальники, як правило, збирають свої пропозиції разом: наприклад, у програмі Red Hat JBoss Middleware є лише один предмет, який включає WildFly та сервери EAP Server, а також JBoss AS.

Перший досвід розробника з Eclipse може викликати неспокій, навіть заплутати. Це тому, що ваше перше завдання - пристосуватись до концептуальної архітектури робочих просторів, перспектив та поглядів Eclipse, функції яких визначаються тим, які плагіни ви встановили. Наприклад, для розробки сервера Java, ви, ймовірно, використовуєте перспективи перегляду Java, Java EE та Java; перегляд пакета провідника; перспектива налагодження; командна перспектива синхронізації; веб-інструменти; перспектива розвитку бази даних; і перспектива налагодження бази даних. На практиці все це почне мати сенс, як тільки ви відкриєте потрібні погляди.

У Eclipse часто існує декілька способів виконати задане завдання. Наприклад, ви можете переглядати код за допомогою програми провідника та/або перспективи перегляду Java; яку ви обираєте - це питання смаку та досвіду.

Підтримка пошуку Java дозволяє знаходити декларації, посилання та входження Java-пакетів, типів, методів та полів. Ви також можете скористатися швидким доступом для пошуку та швидкого перегляду для спливання таких речей, як контури класу.

Додавання методів та генерування класів підтримується анотаціями про помилки та вмістом. З шаблонів коду можна генерувати загальні шаблони коду, а Eclipse може автоматично генерувати та упорядковувати

ваші заяви про імпорт. Рефакторинг Java в Eclipse підтримує 23 операції, починаючи від загальних операцій з перейменування і закінчуючи більш незрозумілими перетвореннями прямо з книги Мартіна Фаулера . Рефакторинг може виконуватися не тільки в інтерактивному режимі, але і за допомогою сценаріїв рефакторингу.

Eclipse підтримує налагодження як локально, так і віддалено, припускаючи, що ви використовуєте JVM, який підтримує віддалену налагодження. Налагодження досить стандартне: ви зазвичай встановлюєте точки перерви, а потім переглядаєте змінні на вкладці перспективи налагодження. Ви, звичайно, можете переглядати свій код і оцінювати вирази.

Eclipse має велику допомогу та документацію різного віку, валюти та корисності. Незвично виявити, що в документацію входять зображення, які не відповідають поточній версії програмного забезпечення, або що натискання клавіш для вашої операційної системи відрізняються від тих, що викликаються в довідці. Боюся, це одна з найпоширеніших проблем з проектами з відкритим кодом: документація може затримувати програмне забезпечення на місяці чи навіть роки. Оскільки екосистема настільки велика, Eclipse має більшу частку, ніж частка документації. (рисунок 1.12).

Рисунок 1.12 – Eclipse

NetBeans Java IDE почав свою життя в якості студента університету проекту в Празі в 1996 році, став комерційним продуктом в 1997 році, була куплена Sun в 1999 році, і був випущений з відкритим вихідним кодом у 2000

році була недавно подарована Apache, і був прийнятий як інкубаційний проект.

Oracle перетворює код на шматки; початкове падіння було лише кодом Java SE. Вам потрібно використовувати 8.2 бібліотеки, щоб отримати інші плагіни, до яких ви могли звикнути. Потрясіння в NetBeans приземляються під час передачі може пояснити, чому NetBeans 9 ще не підтримує JUnit 5: на даний момент інтеграція знаходиться лише на етапі контуру .

Мовний редактор NetBeans виявляє помилки під час введення тексту та допомагає вам зі спливаючими документаціями та заповненням смарт-коду. Здається, це робиться з меншими помітними паузами, ніж Eclipse, хоча трохи більше, ніж IntelliJ IDEA. NetBeans також пропонує повний спектр інструментів рефакторингу, які дозволять вам реструктурувати код, не порушуючи його; виконує аналіз вихідного коду; і пропонує широкий набір підказок для швидкого виправлення або покращення коду. NetBeans включає інструмент для розробки графічних інтерфейсів Swing, раніше відомий як "Project Matisse".

Інструмент Inspect & Transform дозволяє запускати перевірки по всій кодовій базі, автоматично встановлюючи код. Особисто я завжди переконуюсь, що я зареєструвався у всьому своєму коді і успішно запускаю всі мої тестові одиниці, перш ніж запускати інструменти, які можуть внести значні зміни; Мене неодноразово спалювали автоматичні «виправлення», що викликають регресії.

NetBeans має гарну вбудовану підтримку для Maven and Ant та плагін для Gradle. Я був радий дізнатися, що існуючі проекти Maven зараз трактуються як "рідні", тобто ви просто відкриваєте їх, а не імпортуєте. NetBeans також включає в себе сексуальний (і корисний) графік для залежностей від Мейвена.

Налагоджувач Java NetBeans хороший, хоч і звичайний. Окремий візуальний відладчик дозволяє робити знімки графічного інтерфейсу та

візуально досліджувати графічний інтерфейс додатків JavaFX та Swing. Профілер NetBeans дуже приємний для розуміння як процесора, так і використання пам'яті та має хороші інструменти для пошуку витоків пам'яті (рисунок 1.13).

Рисунок 1.13 – NetBeans IDE

IntelliJ IDEA, провідна Java IDE як за функціями, так і за ціною, випускається у двох випусках: безкоштовне видання Community та платне видання Ultimate, яке має додаткові можливості.

Видання Community призначене для розробки JVM та Android. Він підтримує Java, Kotlin, Groovy та Scala; Android; Maven, Gradle і SBT; і Git, SVN, Mercurial, CVS та TFS.

Видання Ultimate, призначене для розробки Інтернету та підприємств, підтримує Perforce на додаток до інших систем управління версіями; підтримує JavaScript і TypeScript; підтримує Java EE, Spring, GWT, Vaadin, Play, Grails та інші рамки; і включає інструменти бази даних та підтримку SQL.

Ідея полягає в тому, що комерційне (Ultimate) видання заробить своє місце на робочому столі професіонала, виправдовуючи платну підписку за рахунок підвищення продуктивності програміста. Якщо ви заробляєте 50 тис.

Доларів США – 100 тис. Доларів на рік як розробник Java, це не потребує значного підвищення продуктивності, щоб отримати швидку рентабельність інвестицій на 500 доларів США на рік підписки на бізнес IDEA. У наступні роки ціна знижується для підприємств, значно нижча для стартапів та приватних осіб, і є безкоштовною для студентів, викладачів, «чемпіонів Java» та розробників з відкритим кодом.

IntelliJ запрошує IDEA для глибокого ознайомлення з вашим кодом, ергономікою розробника, вбудованими інструментами для розробників та досвідом програмування поліглотів. Давайте детально ознайомимося з інформацією про те, що означають ці функції та як вони можуть вам допомогти.

Забарвлення синтаксису та просте завершення коду наведено для редакторів Java. IDEA виходить за рамки цього, щоб забезпечити "розумне завершення", тобто означає, що він може спливати список найбільш релевантних символів, застосовних у поточному контексті. Вони класифікуються за вашою особистою частотою використання. "Завершення ланцюга" йде глибше і відображає список застосованих символів, доступних за допомогою методів або одержувачів в поточному контексті. IDEA також доповнює статичні члени або константи, автоматично додаючи необхідні заяви про імпорт. У всіх доповненнях коду IDEA намагається відгадати тип символу виконання, уточнити його вибір із цього пункту та додати вказівки класів за потребою.

Код Java часто містить інші мови як рядки. IDEA може вводити фрагменти SQL, XPath, HTML, CSS та / або коду JavaScript в літературний рядок Java String. З цього приводу він може рефакторировать код на кількох мовах; наприклад, якщо ви перейменовуєте клас у операторі JPA, IDEA оновить відповідний вираз сутності та вирази JPA.

Коли ви переробляєте фрагмент коду, одна з речей, яку ви зазвичай хочете зробити, - це також рефактор всіх дублікатів цього коду. IDEA

Ultimate може виявити дублікати та подібні фрагменти і застосувати також рефакторинг до них.

IntelliJ IDEA аналізує ваш код під час завантаження та під час введення. Він пропонує інспекції, щоб вказати на можливі проблеми та, за бажанням, список швидких виправлень виявленої проблеми.

IntelliJ розробив IDEA, маючи на увазі творчий потік розробника - він же "перебуває в зоні". Вікно інструмента "Проект", показане ліворуч на рисунку 1, зникає з виду простим клацанням миші, щоб ви могли сконцентруватися на редакторі коду. У всьому, що ви хочете зробити під час редагування, є ярлик на клавіатурі, включаючи відображення визначень символів у спливаючому вікні. У той час як вивчення ярликів вимагає часу та практики, вони з часом стають другими. Навіть не знаючи ярликів, розробник може навчитися легко та швидко використовувати IDEA.

Дизайн відладчика IDEA особливо приємний. Змінні значення відображаються прямо у вікні редактора поруч із відповідним вихідним кодом. Коли стан змінної змінюється, змінюється і колір її виділення.

IntelliJ IDEA надає єдиний інтерфейс для більшості основних систем управління версіями, включаючи Git, SVN, Mercurial, CVS, Perforce та TFS. Ви можете виконати всі керування змінами прямо в IDE. Під час тестування IDEA я хотів, щоб остання зміна блоку вихідного коду відображалася у вікні редактора як примітка (як це робиться у Visual Studio). Як виявляється, для цього є плагін .

IDEA також інтегрує інструменти побудови, тестові бігуни та засоби покриття, а також вбудоване вікно терміналу. IntelliJ не має власного профілера, але він підтримує декілька сторонніх профілів за допомогою плагінів. До них відносяться YourKit, створений колишнім провідним розробником IntelliJ, і VisualVM, яка є перепакованою версією профілера NetBeans.

Налагодження Java може бути болем, коли загадкові речі трапляються в класах, для яких у вас немає вихідного коду. IDEA поставляється з декомпілятором для цих випадків.

Програмування сервера Java часто передбачає роботу з базами даних, тому IDEA Ultimate включає інструменти бази даних SQL та NoSQL. Якщо вам потрібно більше, спеціальна SQL IDE (DataGrip) доступна як частина підписки на всі продукти, яка лише трохи дорожча, ніж підписка IDEA Ultimate.

IntelliJ IDEA підтримує всі основні сервери додатків JVM і може розгортатися на серверах та налагоджувати їх, виправляючи головну больову точку для розробників Enterprise Java. IDEA також підтримує Docker через плагін, який додає вікно інструмента Docker. (Якщо говорити про плагіни, то IntelliJ їх дуже багато .)

IDEA розширила підтримку кодування для Spring, Java EE, Grails, Play, Android, GWT, Vaadin, Thymeleaf, Android, React, AngularJS та інших рамок. Не всі ці рамки Java. На додаток до Java, IDEA розуміє багато інших мов, зокрема Groovy, Kotlin, Scala, JavaScript, TypeScript та SQL. Якщо вам потрібно більше, на даний момент існують сотні плагінів мови IntelliJ, включаючи плагіни для R, Elm, Go, Rust та D (рисунок 1.14). [14]

Рисунок 1.14 – IntelliJ IDEA

Проаналізувавши згадані середовища розробки, вибір було зупинено на Eclipse, так як саме це середовище розробки має велику кількість модулів, що дозволить з легкістю застосувавати їх при розробці самого додатку.

Виберемо тепер тип графічного інтерфейсу.

Swing – це інструментарій для віджетів GUI для Java. Це є частиною Oracle «Java Foundation Classes (JFC) – з API для надання графічного інтерфейсу користувача (GUI) для програм Java.

Swing був розроблений, щоб забезпечити більш досконалий набір компонентів графічного інтерфейсу, ніж попередній Інструментарій абстрактних вікон (AWT) . Swing надає зовнішній вигляд, що імітує зовнішній вигляд декількох платформ, а також підтримує підключається зовнішній вигляд, що дозволяє програмам виглядати і відчувати себе не пов'язаними з базовою платформою. Він має більш потужні та гнучкі компоненти, ніж AWT. Окрім знайомих компонентів, таких як кнопки, прапорці та мітки, Swing пропонує декілька вдосконалених компонентів, таких як панель з вкладками, панелі прокрутки, дерева, таблиці та списки.

На відміну від компонентів AWT, компоненти Swing не реалізовані кодом, визначеним платформою. Натомість вони написані повністю на Java і тому не залежать від платформи. Термін "легкий" використовується для опису такого елемента.

У Internet Foundation Classes (IFC) була графічна бібліотека для Java спочатку розроблений Netscape Communications Corporation і перший випущений 16 грудня 1996 года 2 квітня 1997 року, Sun Microsystems і Netscape Communications Corporation оголосили про свій намір включити МФК з іншими технологіями в формувати класи Java Foundation. «Класи фундації Java» пізніше були перейменовані на «Swing».

Swing представив механізм, який дозволив змінювати зовнішній вигляд кожного компонента програми, не змінюючи коду програми. Впровадження підтримки підключеного зовнішнього вигляду дозволяє компонентам Swing імітувати зовнішній вигляд нативних компонентів, зберігаючи при цьому переваги незалежності платформи. Спочатку розповсюджувався як окремо завантажувана бібліотека, Swing був включений до складу стандартної версії Java з випуску 1.2. Класи Swing та компоненти містяться в ієрархії javax.swing пакетів.

Swing – це незалежний від платформи графічний інтерфейс GUI для «модельного перегляду-контролера» для Java, який слід за однопотоковою моделлю програмування. Крім того, цей фреймворк забезпечує шар абстракції між структурою коду та графічною презентацією GUI на базі Swing.

Swing не залежить від платформи, оскільки він повністю написаний на Java. Повну документацію для всіх класів Swing можна знайти в посібнику Java API для версії 6 або специфікації API платформи Java Platform Standard Edition 8 для версії 8.

Swing – це високомодульна архітектура, яка дозволяє «підключати» різні спеціалізовані реалізації заданих рамкових інтерфейсів: Користувачі можуть надати власну власну реалізацію цих компонентів, щоб замінити виконання за замовчуванням за допомогою механізму успадкування Java.

Swing – це компонентна основа , компоненти якої в кінцевому підсумку походять з класу javax.swing.JComponent . Об'єкти асинхронно розгортають вогневі події, мають пов'язані властивості та реагують на документований набір методів, характерних для компонента. Компоненти Swing – це компоненти Java Beans, які відповідають специфікаціям архітектури компонентів Java Beans.

Велика залежність Swing від механізмів виконання та непрямих моделей композиції дозволяє йому реагувати під час виконання роботи на фундаментальні зміни в налаштуваннях. Наприклад, додаток на базі Swing здатний гаряче змінювати свій інтерфейс користувача під час виконання. Крім того, користувачі можуть забезпечити власну зовнішність та реалізацію, що дозволяє рівномірно змінити зовнішній вигляд існуючих додатків Swing без програмних змін коду програми.

Високий рівень гнучкості Swing виражається в властивій йому здатності перекривати управління GUI рідної операційної системи (OS) для відображення себе. Swing «малює» свої елементи керування за допомогою API 2D Java, а не викликає набір інструментальних інструментів

користувальницького інтерфейсу. Таким чином, компонент Swing не має відповідного вбудованого інтерфейсу інтерфейсу інтерфейсу ОС, і він може візуалізувати будь-яким способом, який можливий із базовими графічними графічними інтерфейсами.

Однак у своїй суті кожен компонент Swing покладається на контейнер AWT, оскільки (Swing's) JComponentрозширює (AWT's) контейнер. Це дозволяє Swing підключатися до рамки управління графічним інтерфейсом ОС, включаючи вирішальне відображення пристрою/екрана та взаємодію користувачів, наприклад, натискання клавіш або рухи миші. Swing просто "транспонує" власну семантику (для ОС-агностики) над базовими (для ОС) компонентами. Так, наприклад, кожен компонент Swing малює своє зображення на графічному пристрої у відповідь на виклик компонента.paint (), який визначений у (AWT) Container. Але на відміну від компонентів AWT, які делегували картину своєму власному ОС «важкої ваги», компоненти Swing відповідають за власне візуалізацію.

Ця транспозиція та розв'язка не є просто візуальною, а поширюється на управління Swing та застосування власної незалежної ОС семантики для подій, запущених у межах її ієрархій стримування компонентів. Взагалі кажучи, архітектура Swing делегує завдання відображення різних ароматів семантики GUI ОС на просту, але узагальнену схему для контейнера AWT. Спираючись на цю узагальнену платформу, вона формує власну багату і складну семантику GUI у вигляді JComponentмоделі.

Бібліотека свінгу робить інтенсивне використання Model/View/Controller програмного забезпечення шаблону проектування, який концептуально відокремлює дані просматріваемога від управління користувальницького інтерфейсу , через який він переглянутий. Через це більшість компонентів Swing мають асоційовані моделі (які визначені з точки зору інтерфейсів Java), і програмісти можуть використовувати різні реалізації за замовчуванням або надавати свої власні. Рамка забезпечує реалізацію модельних інтерфейсів за замовчуванням для всіх її конкретних компонентів.

Типове використання фреймворку Swing не вимагає створення користувацьких моделей, оскільки фреймворк надає набір реалізацій за замовчуванням, які прозоро, за замовчуванням, асоціюються з відповідним JComponentдочірнім класом у бібліотеці Swing Взагалі, лише складні компоненти, такі як таблиці, дерева та іноді списки, можуть вимагати реалізації спеціальної моделі навколо структур даних, що стосуються додатків. Щоб добре зрозуміти потенціал, який дозволяє архітектура Swing, врахуйте гіпотетичну ситуацію, коли спеціальні моделі таблиць і списків обгортки над DAO та/або Послуги EJB.

Зазвичай об'єкти моделей Swing відповідають за надання стислого інтерфейсу, що визначає запущені події та доступні властивості (концептуальної) моделі даних для використання асоційованим JComponent. Зважаючи на те, що загальна модель MVC є слабко пов'язаною схемою співвідношення об'єктних об'єктів, модель надає програмні засоби для приєднання слухачів подій до об'єкта моделі даних. Зазвичай ці події є орієнтованими на модель (наприклад: подія "вставлений рядок" у модель таблиці) і відображаються спеціалізацією JComponent у значущу подію для компонента GUI.

Наприклад, модель JTableмає назву, TableModelяка описує інтерфейс для того, як таблиця отримає доступ до табличних даних. Реалізація цього за замовчуванням працює на двовимірному масиві.

Компонент перегляду Swing JComponent – це об'єкт, який використовується для графічного представлення концептуального управління графічним інтерфейсом. Відмінність Swing як рамки графічного інтерфейсу полягає в його залежності від програмно керованих графічних інтерфейсів управління (на відміну від використання власних елементів управління GUI ОС). До Java 6 Update 10 ця відмінність була джерелом ускладнень при змішуванні елементів керування AWT, які використовують нативні елементи управління, з елементами управління Swing у графічному інтерфейсі (див. Змішування компонентів AWT та Swing).

Нарешті, з точки зору візуального складу та управління, Swing надає перевагу відносним макетам (які задають позиційні відносини між компонентами) на відміну від абсолютних макетів (які задають точне розташування та розмір компонентів). Цей ухил до «рідкого» візуального впорядкування пояснюється його джерелами в операційному середовищі аплетів, що обрамлює дизайн та розробку оригінального набору інструментів Java GUI. (Концептуально цей погляд на управління компонуванням досить схожий на той, який інформує про надання вмісту HTML у браузерах та вирішує той самий набір проблем, що мотивував попереднього) (рисунок 1.15). [15]

Рисунок 1.15 – Swing

JavaFX являє інструментарій для створення кроссплатформенних графічних додатків на платформі Java.

JavaFX дозволяє створювати додатки з багатою насиченою графікою завдяки використанню апаратного прискорення графіки і можливостей GPU.

За допомогою JavaFX можна створювати програми для різних операційних систем: Windows, MacOS, Linux і для самих різних пристроїв:

десктопи, смартфони, планшети, вбудовані пристрої, ТВ. Додаток на JavaFX буде працювати скрізь, де встановлена виконуючого середовища Java (JRE).

JavaFX надає великі можливості в порівнянні з рядом інших подібних платформ, зокрема, в порівнянні зі Swing. Це і великий набір елементів управління, і можливості по роботі з мультимедіа, двомірної і тривимірною графікою, декларативний спосіб опису інтерфейсу за допомогою мови розмітки FXML, можливість стилізації інтерфейсу за допомогою CSS, інтеграція зі Swing і багато іншого.

Історія JavaFX фактично почалася в першій половині 2000-х років, коли розробник на ім'я Кріс Олівер (Chris Oliver), будучи працівником компанії SeeBeyond, розробив для створення графічних інтерфейсів нову мову F3 (Froms Follows Functions). Згодом у 2005 році SeeBeyond була придбана компанією Sun Microsystems (яка на той момент розвивала мову Java до покупки компанією Oracle). F3 був перейменований в JavaFX, а Кріс Олівер продовжив роботу над новою платформою вже в рамках компанії Sun. І в травні 2007 року Sun Microsystems публічно анонсувала нову платформу для створення графічних додатків. А 4 грудня 2008 року вийшов JavaFX 1.0 SDK.

Після придбання Sun Microsystems компанією Oracle в 2010 році була анонсована, а в 2011 році вийшла в реліз версія JavaFX 2.0. У першій версії JavaFX фактично представляв скіптовий мову. У другій версії був повністю змінений підхід. Скриптова мова була прибрана, а платформа була повністю переписана фактично з нуля. Тепер створювати додатки можна було за допомогою будь-якої мови, яка підтримувала JVM. Були додані нові API, інтеграція зі Swing і багато інших речей.

Наступними важливими віхами в розвитку платформи стали версії JavaFX 8 і особливо JavaFX 9, яка вийшла у вересні 2017 року разом з Java 9 і привнесла в платформу модульність. І якщо раніше JavaFX поставлялася разом з Java SE, то зараз JavaFX відокремлена від основної функціональності

Java SE і використовується як окремий модуль. Остання версія фреймворку - JavaFX 12 – вийшла в березні 2019 року.

На даний момент багато розробників називають JavaFX кращи способом для створення графічних додатків за допомогою мови Java, навідміну від AWT і Swing. Також варто відзначити, що для роботи з JavaFX замість Java теоретично можна використовувати будь-яку мову програмування, яка підтримується JVM (рисунок 1.16). [16]

Рисунок 1.16 – JavaFX

Було вибрано JavaFX як інструмент створення графічного інтерфейсу користувача, так як цей інструмент в багатьох аспектах простий та легкий для розгортання.

1.3.2 Вибір СУБД

Архітектуру бази даних було обрано реляційну.

Реляційна база даних – це тип бази даних. Він використовує структуру, яка дозволяє нам ідентифікувати та отримувати доступ до даних стосовно іншого фрагмента даних у базі даних. Дані в реляційній базі даних організовані в таблиці.

Таблиця являє собою набір елементів даних (значення).

Частину даних у файлі називають записом.

Кожен елемент запису називається полем.

Таблиця, запис, поле, рядок і стовпець

Один фрагмент даних або запис називається рядком .

Кожен елемент або поле називається стовпцем .

Первинний ключ, унікальний та нульовий

Полю часто відводиться важлива роль у базі даних, коли це відбувається, ми називаємо це поле Первинним ключем . У цьому прикладі Код товару є первинним ключем. Більше інформації про це в наступному розділі.

Унікальне значення є значенням, яке не може бути повторено (назва продукту , як, ви не повинні мати два продукти з такою ж назвою).

Нульовим є відсутність значення (як видно вище у «Зауваженнях», де є порожні значення). Деякі поля можуть бути нульовими (залежно від бази даних).

Ключ: один або кілька стовпців таблиці бази даних, які використовуються для сортування та/або ідентифікації рядків у таблиці. наприклад, якщо ви сортували людей за зарплатою на місцях, то ключовим є поле заробітної плати.

Первинний ключ – це одне або кілька полів, які однозначно ідентифікують рядок у таблиці. Первинний ключ не може бути нульовим (порожнім). Первинний ключ індексується (докладніше про індекс пізніше).

Зовнішній ключ – це відношення між стовпцями у двох таблицях баз даних (одна з яких індексується), призначених для забезпечення послідовності даних.

Композитний ключ: ключ, що складається з одного або декількох стовпців. Первинний ключ можна сформувати за допомогою полів (хоча і не дуже доцільно).

Природний ключ: складений первинний ключ, який складається з атрибутів (полів), які вже існують у реальному світі (наприклад, ім'я, прізвище, номер соціального страхування).

Сурогатний ключ: ключ, який генерується внутрішньо (як правило, цільове значення автоматичного збільшення), який не існує в реальному світі (наприклад, ідентифікатор, який служить для ідентифікації запису, але нічого іншого).

Ключ кандидата: стовпчик або набір стовпців у таблиці, яка дозволяє однозначно ідентифікувати будь-яку запис бази даних, не посилаючись на будь-які інші дані. Кожна таблиця може мати один або декілька ключових ключів, але один ключ-кандидат є унікальним (первинний ключ).

Складений ключ: ключ, що складається з двох або більше полів, що однозначно описують рядок у таблиці. Різниця між складовою та кандидатом полягає в тому, що всі поля в складеному ключі є сторонніми ключами; у ключі-кандидаті одне або декілька полів можуть бути сторонніми ключами (але це не є обов'язковим).

Нормалізація – обробка таблиць даних із реального світу для реляційної бази даних, дотримуючись серії кроків. Для правильного управління реляційною базою даних необхідна нормалізація даних. Нормалізація використовується в основному для двох цілей:

- Усунення зайвих (марних) даних.
- Забезпечення залежності даних має сенс, тобто дані логічно зберігаються.

Ненормована форма

Перша нормальна форма створюється з цієї таблиці. Усі атрибути, визначені вами для даної сутності, ймовірно, згруповані в плоску структуру. Тут відбувається процес нормалізації для впорядкування атрибутів.

Перша нормальна форма

Щоб таблиця була у першій звичайній формі, вона повинна дотримуватися наступних 4 правил:

– Він повинен мати лише поодинокі (атомні) значення атрибутів / стовпців : Це означає, наприклад, що фрукт не повинен знаходитися в базі даних з двома назвами.
– Значення, що зберігаються в стовпці, повинні бути одного домену : це більше правило "Звичайний сенс". У кожному стовпці значення, що зберігаються, повинні бути одного виду або типу.
– Усі стовпці таблиці повинні мати унікальні імена : Це правило передбачає, що кожен стовпець таблиці має унікальну назву. Це дозволяє уникнути плутанини під час отримання даних або виконання будь-якої іншої операції зі збереженими даними.
– Порядок збереження даних не має значення : Це правило говорить про те, що порядок зберігання даних у вашій таблиці не має значення.

Важливо ще пам’ятати, що:

– Основні атрибути: частини ключа-кандидата даної реляційної таблиці.
– Непримітні атрибути: не є частиною ключа ключа.

Друга нормальна форма

– Він повинен бути у формі Першої Нормальної.
– Вона не повинна мати часткової залежності . Якщо непростий атрибут відношення отримується лише частиною складеного ключа-кандидата, то така залежність визначається як часткова залежність.
– Залежність: Коли вам потрібно використовувати первинний ключ, щоб отримати певне значення (наприклад, ваше ім'я, щоб знати ваш вік).

Третя нормальна форма

– Це у другій нормальній формі.

– Він не має перехідної залежності . Якщо непростий атрибут відношення отримується іншим атрибутом, який не є простим, або комбінацією частини ключового ключа разом із атрибутом, який не є простим, то така залежність буде визначена як перехідна залежність. [17]

Як зв’язок з базою даних використовується JDBC.

Java Database Connectivity - це стандартний API для незалежного з'єднання мови програмування Java з різними базами даних.

JDBC вирішує наступні завдання:

– Створення з'єднання з БД.

– Створення SQL виразів.

– Виконання SQL - запитів.

– Перегляд і модифікація отриманих записів.

Якщо говорити в цілому, то JDBC – це бібліотека, яка забезпечує цілий набір інтерфейсів для доступу до різних БД.

Для доступу до кожної конкретної БД необхідний спеціальний JDBC – драйвер, який є адаптером Java – додатки до БД.

JDBC підтримує як 2-шарову, так і 3-шарову модель роботи з БД, але в загальному вигляді, JDBC складається з двох шарів.

JDBC API забезпечує з'єднання "додаток - JDBC Manager".

JDBC Driver API забезпечує з'єднання "JDBC Manager - драйвер".

JDBC API використовує менеджер драйверів і спеціальні драйвери БД для забезпечення підключення до різних баз даних.

JBDC Manager перевіряє відповідність драйвера і конкретної БД. Він підтримує можливість використання декількох драйверів одночасно для одночасної роботи з декількома видами БД.

JDBC API складається з наступних елементів:

– Менеджер драйверів (Driver Manager). Цей елемент управляє списком драйверів БД. Кожній запит на з'єднання вимагає відповідного драйвера. Перше збіг дає нам з'єднання.
– Драйвер (Driver). Цей елемент відповідає за зв'язок з БД. Працювати з ним нам доводиться вкрай рідко. Замість цього ми частіше використовуємо об'єкти DriverManager, які керують об'єктами цього типу.
– З'єднання (Connection). Цей інтерфейс забезпечує нас методами для роботи з БД. Всі взаємодії з БД відбуваються виключно через Connection.
– Вираз (Statement). Для підтвердження SQL-запитів ми використовуємо об'єкти, створені з використанням цього інтерфейсу.
– Результат (ResultSet). Примірники цього елемента містять дані, які були отримані в результаті виконання SQL - запиту. Він працює як ітератор і "пробігає" по отриманим даним.
– Винятки (SQL Exception). Цей клас обробляє всі помилки, які можуть виникнути при роботі з БД. [18]

Розглянемо популярні реляційні бази даних (БД) та системи управління базами даних (СУБД).

MySQL («Мій SQL») є відкритим вихідним код реляційної системи управління базами даних (СКБД). Його назва – це поєднання "My", прізвище дочки співзасновника Майкла Віденуса та «SQL», абревіатура для Structured Query Language .

MySQL – це безкоштовне програмне забезпечення з відкритим кодом за умовами Загальної публічної ліцензії GNU , а також доступне під різними власними ліцензіями. MySQL належить та спонсорується шведською компанією MySQL AB , яку придбала компанія Sun Microsystems (зараз корпорація Oracle). У 2010 році , коли Oracle придбала Sun, Widenius

роздвоєний на відкритим вихідним кодом MySQL проект по створенню MariaDB .

MySQL є компонентом стеку програмного забезпечення для веб-додатків LAMP (та інших), що є абревіатурою для Linux, Apache, MySQL, Perl/ PHP/Python. MySQL використовується багатьма веб-програмами, керованими базами даних, включаючи Drupal, Joomla, phpBB та WordPress. MySQL також використовується на багатьох популярних веб-сайтах, включаючи Facebook, Flickr, MediaWiki, Twitter, та YouTube.

MySQL написано на C та C++. Його SQL-аналізатор написаний на уасс, але він використовує домашній лексичний аналізатор. MySQL працює на багатьох системних платформах, включаючи AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos і Tru64. Порт MySQL для OpenVMS також існує.

Саме програмне забезпечення сервера MySQL та бібліотеки клієнтів використовують розподіл подвійного ліцензування. Вони пропонуються згідно GPL версії 2, або власницької ліцензії.

Підтримку можна отримати в офіційному посібнику. Додаткова безкоштовна підтримка доступна на різних каналах та форумах IRC. Oracle пропонує платну підтримку через свої продукти MySQL Enterprise. Вони відрізняються за обсягом послуг та ціною. Крім того, існує ряд сторонніх організацій, які надають підтримку та послуги, включаючи MariaDB та Percona.

MySQL отримав позитивні відгуки, багато користувачів кажуть, що це надзвичайно добре зазвичай, і що інтерфейси для розробників є, і документація (не кажучи вже про відгуки в реальному світі через веб-сайти тощо) дуже, дуже добре. Він також був перевірений як швидкий, стабільний і справжній багатокористувацький багатопотоковий сервер бази даних sql. (рисунок 1.17). [19]

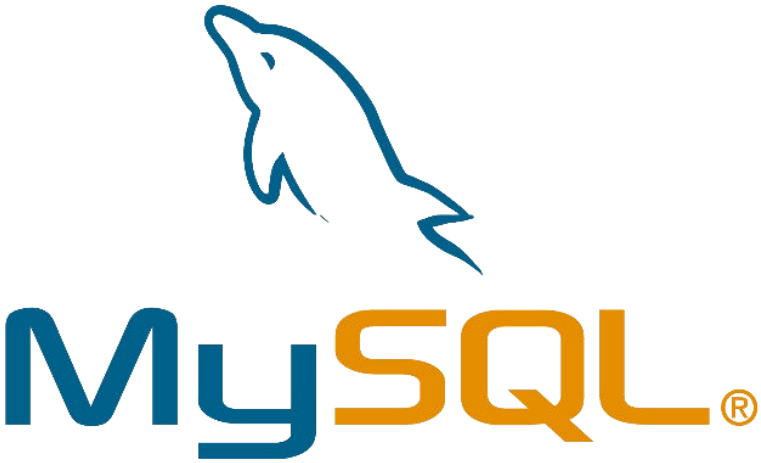


Рисунок 1.17 – MySQL

Якщо за хмарну частина обробки даних відповідає Azure SQL, то за локальну складову платформи Microsoft для зберігання і обробки даних – Microsoft SQL Server 2019.

SQL Server спрощує розгортання, передачу та інтеграцію великих даних.

Рішення для обробки великих даних на основі Kubernetes, вбудоване в SQL Server, дозволяє легко розгорнути кластер великих даних і працювати з ним. Kubernetes забезпечує розгортання сховищ HDFS, реляційного модуля SQL Server і засобів аналітики Spark у вигляді контейнерів в рамках одного зручного пакету.

До складу SQL Server 2019 входять Spark і HDFS, які дозволяють виконувати читання і запис безпосередньо в HDFS, використовуючи SQL Server або Spark. Архітектура Kubernetes забезпечує гнучке масштабування обчислювальних потужностей і сховищ за запитом

2. Інтеграція структурованих і неструктурованих даних.

Сьогоднішні обсяги даних роблять нерозумним і невигідним конвертацію всіх доступних даних в реляційні таблиці для зберігання в СУБД. Ще 2 роки тому Microsoft представила технологію PolyBase, що дозволяє примірнику SQL Server обробляти запити Transact-SQL, які

звертаються до даних Hadoop і об'єднувати дані з Hadoop і SQL Server. У SQL Server зовнішня таблиця або зовнішнє джерело даних забезпечує з'єднання з Hadoop, віртуалізіруя зовнішні джерела даних без необхідності їх прямого імпорту в реляційну базу, і потім дозволяє звертатися до цих даних із запитами.

Таким чином, дані накопичуються в своєму природному форматі, не обов'язково реляционном, але можуть бути представлені у вигляді віртуальної таблиці. Віртуалізація дозволяє інтегрувати дані різного формату, з різнорідних джерел і місць зберігання без їх реплікації і переміщення, створюючи єдину віртуальну матрицю даних.

3. Висока продуктивність.

Не перший рік Microsoft підтверджує високу продуктивність SQL Server транзакційними тестів і тестами продуктивності сховищ даних. Версія 2019 відзначена відмінними результатами в наступних тестах:

4. Підтримка постійної пам'яті (PMEM).

Постійна пам'ять (Persistent Memory, PMEM) - це швидка пам'ять, що володіє можливістю зберігати дані після відключення живлення. Вона дозволяє обробляти дані in-memory, позбавляючи від необхідності передавати дані по каналах передачі і прискорюючи обробку запитів на 30% для інтенсивних робочих навантажень введення-виведення.

Будь-який файл SQL Server, поміщений на пристрій PMM, тепер доступний безпосередньо, минаючи стек зберігання операційної системи, використовуючи ефективні операції memcpy.

5. Гібридна транзакційна / аналітична обробка (Htape).

Модель HTAP дозволяє одночасно здійснювати операційні транзакції і аналітику на одних і тих же даних в одній і тій же пам'яті, також реалізуючи підхід in memory.

6. Інтелектуальна обробка запитів

Параллелизация запитів і поліпшене масштабування частих запитів завдяки механізмам інтелектуальної обробки запитів роблять продуктивність

значно вище. Відкладене компіляція табличних змінних більш ніж на 50% прискорює обробку запитів.

7. Безпека і відповідність вимогам.

Захист конфіденційних даних за допомогою технології Always Encrypted з захищеними анклавами. Шифрування на місці дозволяє виконувати криптографічні операції з конфіденційними даними без їх переміщення за межі бази даних.

Криптографічні операції включають в себе шифрування стовпців, і ці операції тепер можна виконувати за допомогою Transact-SQL, вони не вимагають переміщення даних з бази даних. Усередині захищених анклавів підтримуються всі повнофункціональні обчислення, включаючи зіставлення і порівняння діапазонів, що значно розширює можливості їх застосування

Технологія Always Encrypted з захищеними анклавами доступна в Windows Server 2019.

8. Azure Data Studio.

Azure Data Studio (колишній SQL Operations Studio) – це спрощене кроссплатформне графічний засіб управління і редактор коду. Дозволяє створювати запити до реляційних і нереляційних баз даних з підтримкою різних операційних систем і джерел даних. Azure Data Studio дозволяє підключатися до SQL Server локально і в хмарі, в Windows, macOS і Linux.

9. Вибір ОС і контейнерів.

SQL Server 2019 відрізняється гнучкістю щодо вибору платформи, мови програмування та засоби доставки.

Підтримка Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Ubuntu і Windows.

Контейнери Docker для Linux і Windows. Установка з вбудованою підтримкою інструментів Linux: Yum lnstall, Apt-Get і Zypper.

Один і той же рівень абстракції з SQL Server на Linux.

Можливість використання R, Python і Java при роботі з T-SQL . Тепер розширення мови Java є для виконання коду Java в SQL Server.

10. Інтелектуальний аналіз даних.

Розвиток SQL Server пошло по шляху інтеграції з іншими аналітичними платформами, зокрема Spark, яка включена тепер в поставку SQL Server.

Spark є дуже популярним інструментом для машинного навчання, для просунутої аналітики, має ефективну in memory машину. І все це інтегровано з SQL, який дуже ефективний для візуалізації аналітики.

Правильний аналіз і ефективне представлення результатів безпосередньо впливає на ефективність аналізу даних і можливість приймати на їх основі управлінські рішення (рисунок 1.18). [20]

Рисунок 1.18 – Microsoft SQL Server

SQLite – це внутрішня бібліотека, яка реалізує автономну, безсерверну, нульову конфігурацію, транзакційний двигун бази даних SQL. Код для SQLite знаходиться у відкритому доступі і тому є вільним для використання для будь-яких цілей, комерційних чи приватних. SQLite – це найбільш широко розгорнута база даних у світі з дуже великою кількістю додатків включаючи декілька гучних проектів.

SQLite – це вбудований двигун бази даних SQL. На відміну від більшості інших баз даних SQL, у SQLite немає окремого серверного процесу. SQLite читає і записує безпосередньо у звичайні файли диска.

Повна база даних SQL з кількома таблицями, індексами, тригерами та переглядами міститься в одному файлі диска. Формат файлу бази даних є кросплатформенним - ви можете вільно копіювати базу даних між 32-бітовою та 64-бітовою системами або між архітектурами big-endian та little-endian . Ці функції роблять SQLite популярним вибором як Формат файлу додатків . Файли баз даних SQLite – це рекомендований формат зберігання Бібліотекою Конгресу США. Не думайте про SQLite не як про заміну Oracleале замість fopen.

SQLite – це компактна бібліотека. Якщо ввімкнути всі функції, розмір бібліотеки може бути менше 600 Кбіт, залежно від цільової платформи та налаштувань оптимізації компілятора. (64-бітний код більший. А деякі оптимізації компілятора, такі як агресивна функція вбудовування функцій та розкручування циклу, можуть призвести до того, що об'єктний код буде значно більшим) Існує компроміс між використанням пам'яті та швидкістю. Зазвичай SQLite працює швидше, чим більше пам'яті ви йому надаєте. Тим не менш, продуктивність, як правило, досить хороша навіть в умовах низької пам’яті. Залежно від способу його використання, SQLite може бути швидшим, ніж прямий вхід/вивід файлової системи .

SQLite дуже ретельно перевіряється перед кожним випуском і має репутацію дуже надійної. Більшість вихідних кодів SQLite присвячена виключно тестуванню та верифікації. Автоматизований набір тестів запускає мільйони та мільйони тестових випадків із залученням сотень мільйонів окремих SQL-висловлювань та досягає 100% охоплення галузевим тестом . SQLite витончено реагує на збої в розподілі пам'яті та помилки вводу / виводу диска. Операції є за правилами ACID навіть якщо вони перервані збою системи або відключенням електроживлення. Все це підтверджено автоматизованими тестами за допомогою спеціальних тестів, що імітують збої в системі. Звичайно, навіть при всьому цьому тестуванні все ж є помилки. Але на відміну від деяких подібних проектів (особливо

комерційних конкурентів), SQLite відкрито і чесно ставиться до всіх помилок і надає списки помилок і хронології хронології змін коду.

База коду SQLite підтримується міжнародною командою розробників, які працюють над SQLite повний робочий день. Розробники продовжують розширювати можливості SQLite та підвищувати його надійність та продуктивність, зберігаючи зворотну сумісність із опублікованими специфікаціями інтерфейсу, синтаксисом SQL та форматом файлу бази даних. Вихідний код абсолютно безкоштовний для всіх, хто цього хоче, але також доступна професійна підтримка.

Проект SQLite розпочато 2000-05-09. Майбутнє завжди важко передбачити, але мета розробників полягає у підтримці SQLite до 2050 року. Дизайнерські рішення приймаються з цією метою (рисунок 1.19). [21]

Рисунок 1.19 - SQLite

Порівняльний аналіз наведених вище моделей баз даних та СУБД дозволив вибрати SQLite як модель бази даних, що буде використовуватись в проекті. Вирішальними перевагами SQLite були легке розгортання та під’єднання програми.

1.3.3 Реалізація основних класів та методів

Так як розробка проводиться на мові програмування Java з використанням фреймворку JavaFX в середовищі розробки Eclipse, розробку слід починати з класу Main, як клас, з якого виклику методу main якого

починається робота програми. Основна робота класу Main – виклик головного вікна програми. Якщо детальніше, то виклик вигляду вікна і прив'язка контролера до вікна. Додатково клас Main ще викликає під'єднання до БД, так як програма тісно працює з базою даних (лістинг 1.1).

Лістинг 1.1 – Основна частину класу Main

```
package application;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.time.DayOfWeek;
import java.time.LocalDate;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import java.util.Random;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.stage.Stage;
import model.DB;
import model.Visiting;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.AnchorPane;

public class Main extends Application
{

       public static String dataFormat = "dd.MM.yyyy";


    private void setConnectionToDb()
    {
       try
       {
           String curFolder = new File(".").getCanonicalPath().toString();
           if (new File(curFolder, "students.db").exists())
           {
              DB.setDbLocation(curFolder + File.separatorChar + "students.db");
           } else
           {
              Alert alert = new Alert(AlertType.ERROR);
              alert.setTitle("Вікно помилки");
              alert.setHeaderText(
                     "У теці з програмою не було знайдено базу даних\nза " +
"замовчуванням (файл students.db)");
              alert.setContentText("Будь ласка, перед тим, як використовувати
програму,\n"
                     + "виберіть файл бази даних\nчерез кнопку 'Вибрати файл бази
даних'\n"
                     + "у вкладці 'Управління БД'.");
```

```
            alert.showAndWait();
        }
    } catch (IOException e1)
    {
        e1.printStackTrace();
    }
}

@Override
public void start(Stage stage)
{
    try
    {
        setConnectionToDb();
        AnchorPane root = (AnchorPane)
FXMLLoader.load(getClass().getResource("/view/MainWindow.fxml"));

        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.setTitle("Журнал");
        stage.show();


    } catch (Exception e)
    {
        e.printStackTrace();

        javafx.scene.control.Alert alert = new
javafx.scene.control.Alert(AlertType.ERROR);
        alert.setTitle("Сталася помилка");
        alert.setHeaderText("Помилка:" + e.getMessage());

        StringWriter stackTraceWriter = new StringWriter();
        e.printStackTrace(new PrintWriter(stackTraceWriter));

        alert.setContentText(e.toString() + "\n" + stackTraceWriter.toString());

        alert.showAndWait();
    }
}

public static void main(String[] args)
{
    launch(args);
}

public static void callAllertChangesSaved()
{
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Успіх!");
    alert.setHeaderText(null);
    alert.setContentText("Зміни збережено!");
    alert.showAndWait();
}
}
```

Розглянемо тепер основні класи моделі. Почнімо з класу Student. Даний клас відповідає сутності Студент та працює з даними студента. А саме

дата народження, ім’я і стать. Даний клас має методи, які видають, змінюють дані студента, отримують статистику присутностей та успішності студентів додають та видаляють студентів з групи та багато іншого (рисунок 1.20)

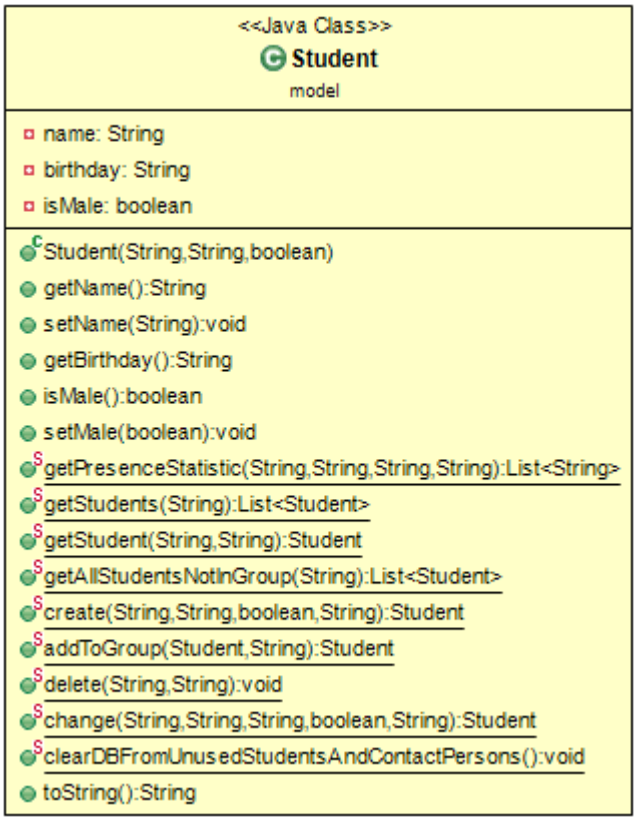


Рисунок 1.20 – Клас Student

Основна частина коду зображена в лістингу 1.2.

Лістинг 1.2 – Основна частина класу Student

```
package model;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class Student
{

    private String name;
    private String birthday;
    private boolean isMale;

    /**
     * @param name
     *            of Student
     * @param birthday
     *            of Student in "yyyy-MM-dd" format. Ex.: "2000-12-30".
     * @param ismale
```

```
     *            true if male, false if female.
     */
    public Student(String name, String birthday, boolean ismale) {
       this.name = name;
       this.birthday = birthday;
       this.isMale = ismale;
    }

    public String getName()
    {
       return name;
    }

    public void setName(String name)
    {
       this.name = name;
    }

    public String getBirthday()
    {
       return birthday;
    }

    /**
     * @return true if student is male, false if female.
     */
    public boolean isMale()
    {
       return isMale;
    }

    /**
     * @param isMale
     *            true if student is male, false if female.
     */
    public void setMale(boolean isMale)
    {
       this.isMale = isMale;
    }

    /**
     * Get presence statistic of student
     *
     * @param studentName
     *            name of student
     * @param curGroupName
     *            name of group
     * @param dates
     *            dates to look between them like "2018-02-10 2018-02-15" or ""
     * @return list of statistics {average marks, count of presences, count of
     *         absence}
     */
    public  static  List<String>  getPresenceStatistic(String  studentName,  String
curGroupName, String mainGroupName,
          String dates)
    {
       List<String> list = new ArrayList<String>();

       String querryGroupPart = "";
       if (curGroupName != "<Bci>")
       {
```

```
            querryGroupPart = " AND group_name = '" + curGroupName + "'";
        }

        String querry1 = "SELECT COUNT(date) "
                + "FROM Visitings_of_students WHERE student_name = '"
                + studentName + "' AND birthdate IN (SELECT birthdate FROM Students "
                + "WHERE studentid IN (SELECT studentid FROM Students_to_groups "
                + "WHERE groupid IN (SELECT groupid FROM Groups WHERE name = '" +
mainGroupName
                + "')))" + querryGroupPart;

        String querry2 = "SELECT COUNT(date) "
                        + "FROM Visitings_of_students WHERE student_name = '"
                        + studentName + "' AND birthdate IN (SELECT birthdate FROM
Students "
                        + "WHERE studentid IN (SELECT studentid FROM Students_to_groups "
                        + "WHERE groupid IN (SELECT groupid FROM Groups WHERE name = '" +
mainGroupName
                        + "'))) AND presence != 0" + querryGroupPart;

        String querry3 = "SELECT AVG(mark) FROM Visitings_of_students "
                + "WHERE student_name = '" + studentName + "' AND birthdate IN (SELECT
birthdate FROM Students "
                + "WHERE studentid IN (SELECT studentid FROM Students_to_groups "
                + "WHERE groupid IN (SELECT groupid FROM Groups WHERE name = '" +
mainGroupName
                + "'))) AND mark != 0" + querryGroupPart;

        if (dates != "")
        {
            String[] splittedDates = dates.split(" ");
            querry1 += " AND date >= '" + splittedDates[0] + "' AND date <= '" +
splittedDates[1] + "'";
            querry2 += " AND date >= '" + splittedDates[0] + "' AND date <= '" +
splittedDates[1] + "'";
            querry3 += " AND date >= '" + splittedDates[0] + "' AND date <= '" +
splittedDates[1] + "'";
        }
        querry1 += ";";
        querry2 += ";";
        querry3 += ";";

        DB db = DB.conn();
        List<String> strList1 = db.select(querry1);
        List<String> strList2 = db.select(querry2);
        List<String> strList3 = db.select(querry3);

        for (String str : strList1)
        {
            String[] split1 = str.split("\n");
            String[] split2 = strList2.get(0).split("\n");
            String[] split3 = strList3.get(0).split("\n");

            list.add(split3[0]);
            list.add(split2[0]);
            list.add(""           +           (Integer.parseInt(split1[0])           -
Integer.parseInt(split2[0])));
        }

        return list;
    }
```

```
    public static List<Student> getStudents(String groupName)
    {
       List<Student> list = new LinkedList<Student>();

       String querry = "SELECT student_name, birthdate, ismale FROM
Students_in_groups WHERE "
             + "group_name = '" + groupName + "' ORDER BY student_name;";

       DB db = DB.conn();
       List<String> strList = db.select(querry);

       for (String str : strList)
       {
           String[] split = str.split("\n");
           boolean isNewStudentMale = Integer.parseInt(split[2]) == 1;
           list.add(new Student(split[0], split[1], isNewStudentMale));
       }
       System.out.println("Size of students list of '" + groupName + "': " +
list.size());
       return list;
    }

    public static Student create(String name, String birthday, boolean isMale, String
groupName)
    {
       Student newStudent = new Student(name, birthday, isMale);

       String querry = "INSERT INTO Students (name, birthdate, ismale) " + "VALUES
('" + newStudent.getName()
             + "', '" + newStudent.getBirthday() + "', " + ((newStudent.isMale()) ?
1 : 0)
             + ");";

       DB db = DB.conn();

       db.executeUpdate(querry);

       querry = "INSERT INTO Students_to_groups (studentid, groupid) VALUES "
                    + "((SELECT studentid FROM Students WHERE name = '" +
newStudent.getName()
                    + "' AND birthdate = '" + newStudent.getBirthday() + "' AND
ismale = "
                    + ((newStudent.isMale()) ? 1 : 0) + " LIMIT 1), (SELECT groupid
FROM Groups "
                    + "WHERE name = '" + groupName + "' LIMIT 1));";

       db.executeUpdate(querry);

       return newStudent;
    }

    public static Student addToGroup(Student student, String groupName)
    {
       String querry = "INSERT INTO Students_to_groups (studentid, groupid) VALUES "
                    + "(SELECT studentid FROM Students WHERE name = '" +
student.getName()
                    + "' AND birthdate = '" + student.getBirthday() + "' AND ismale =
"
                    + ((student.isMale()) ? 1 : 0) + " LIMIT 1), (SELECT groupid FROM
Groups "
```

```
                    + "WHERE name = '" + groupName + "');";

        DB db = DB.conn();
        db.executeUpdate(querry);

        return student;
    }

    public static void delete(String studentName, String groupName)
    {
        String querry = "DELETE FROM Students_to_groups WHERE studentid IN "
                    + "(SELECT studentid FROM Students WHERE name = '" + studentName
                    + "') AND groupid IN (SELECT groupid FROM Groups "
                    + "WHERE name = '" + groupName + "');";

        DB db = DB.conn();

        db.executeUpdate(querry);

        clearDBFromUnusedStudentsAndContactPersons();
    }


    @Override
        public String toString() {
              return name + ", " + birthday + ", " + (isMale ? "хлопчик" :
"дівчинка");
        }
}
```

Клас Group відповідає сутності групи. В даному класі є такі поля, як назва групи, перелік днів тижня, в які проводяться заняття, дати початку та кінця навчання групи та список студентів групи. Також даний клас містить в собі методи роботи з даною сутністю. А саме методи отримання та зміни полей об’єкта, отримання даних групи за іменем з БД, додавання та вилучення студентів групи тощо (рисунок 1.21).

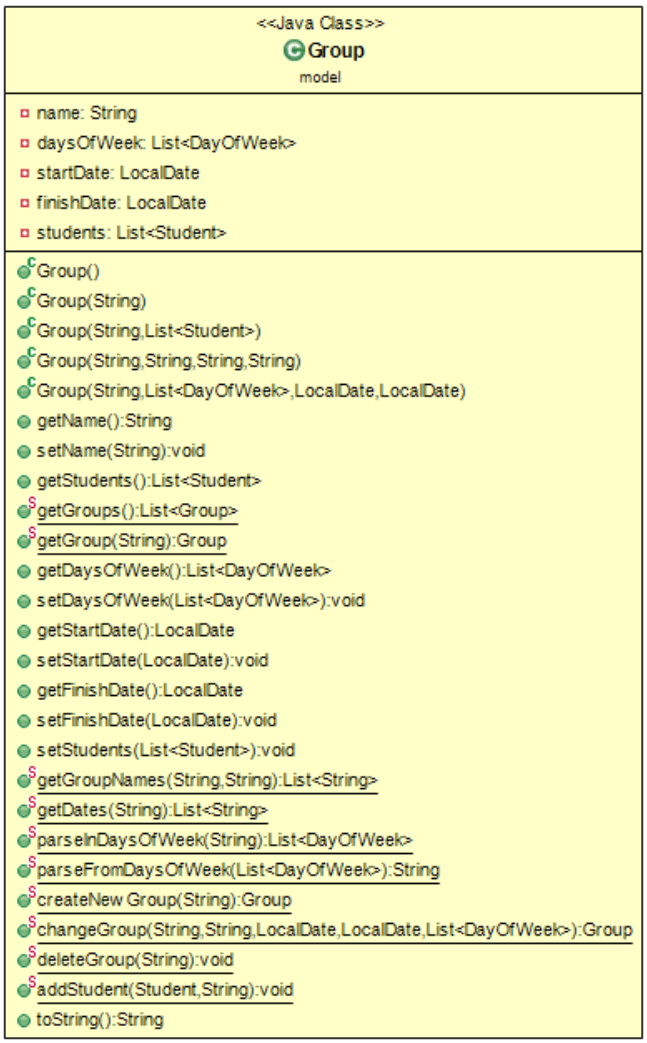


Рисунок 1.21 – Клас Group

Основна частина коду класу Group зображено в лістингу 1.3.

Лістинг 1.3 – Основна частина класу Group

```
package model;

import java.time.DayOfWeek;
import java.time.LocalDate;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;

public class Group
{

    private String name;
    private List<DayOfWeek> daysOfWeek;
    private LocalDate startDate;
    private LocalDate finishDate;
    private List<Student> students;

    public Group() {
```

```
        super();
        this.name = "";
        this.students = new LinkedList<Student>();
        this.startDate = LocalDate.parse("2018-01-01");
        ;
        this.finishDate = LocalDate.parse("2018-01-02");
        this.daysOfWeek = new LinkedList<DayOfWeek>();
    }

    public Group(String name) {
        super();
        this.name = name;

        this.students = new LinkedList<Student>();
    }

    public Group(String name, List<Student> students) {
        super();
        this.name = name;
        this.students = students;
    }

    public Group(String name, String daysOfWeek, String startDate, String finishDate)
{
        super();
        this.name = name;
        this.students = new LinkedList<Student>();
        this.startDate = LocalDate.parse(startDate);
        this.finishDate = LocalDate.parse(finishDate);
        this.daysOfWeek = parseInDaysOfWeek(daysOfWeek);
    }

    public  Group(String  name,  List<DayOfWeek>  daysOfWeek,  LocalDate  startDate,
LocalDate finishDate) {
        super();
        this.name = name;
        this.students = new LinkedList<Student>();
        this.startDate = startDate;
        this.finishDate = finishDate;
        this.daysOfWeek = daysOfWeek;
    }

    public String getName()
    {
        return name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public List<Student> getStudents()
    {
        return students;
    }

    public static List<Group> getGroups()
    {
        List<Group> list = new LinkedList<Group>();
        String querry = "SELECT name, daysOfWeek, startDate, finishDate FROM Groups";
```

```
        DB db = DB.conn();
        List<String> strList = db.select(querry);
        for (String str : strList)
        {
            String[] split = str.split("\n");

            Group newGroup = new Group(split[0], split[1], split[2], split[3]);

            list.add(newGroup);
        }
        System.out.println(list.toString());
        return list;
    }

    public static Group getGroup(String groupName)
    {
        Group group = new Group();
        String querry = "SELECT name, daysOfWeek, startDate, finishDate FROM Groups
WHERE " + "name = '" + groupName
                + "'";
        ;
        DB db = DB.conn();
        List<String> strList = db.select(querry);
        for (String str : strList)
        {
            String[] split = str.split("\n");

            group = new Group(split[0], split[1], split[2], split[3]);

        }
        System.out.println(group.toString());
        return group;
    }

    public List<DayOfWeek> getDaysOfWeek()
    {
        return daysOfWeek;
    }

    public void setDaysOfWeek(List<DayOfWeek> daysOfWeek)
    {
        this.daysOfWeek = daysOfWeek;
    }

    public LocalDate getStartDate()
    {
        return startDate;
    }

    public void setStartDate(LocalDate startDate)
    {
        this.startDate = startDate;
    }

    public LocalDate getFinishDate()
    {
        return finishDate;
    }

    public void setFinishDate(LocalDate finishDate)
    {
```

```
      this.finishDate = finishDate;
    }

    public void setStudents(List<Student> students)
    {
      this.students = students;
    }

    /**
     * Get names of groups of some student
     *
     * @param studentName
     *            - name of student
     * @param groupName
     *            - name of 1 of his group
     * @return list of names of his groups (search students with same name and
     *         birthdays in all groups)
     */
    public static List<String> getGroupNames(String studentName, String groupName)
    {
      List<String> groupNames = new LinkedList<String>();
      String querry = "SELECT group_name FROM Students_in_groups "
            + "WHERE student_name = '" + studentName
            + "' AND birthdate IN (SELECT birthdate FROM Students_in_groups "
            + "WHERE group_name = '" + groupName + "');";

      DB db = DB.conn();
      List<String> strList = db.select(querry);
      for (String str : strList)
      {
          groupNames.add(str.split("\n")[0]);
      }
      return groupNames;
    }

    public static Group createNewGroup(String name)
    {
      Group newGroup = new Group(name, "", LocalDate.now().toString(),
LocalDate.now().toString());
      String querry = "INSERT INTO Groups (name, daysOfWeek, startDate, finishDate)
VALUES ('" + name + "', '', '"
            + newGroup.getStartDate().toString() + "', '" +
newGroup.getFinishDate().toString() + "');";
      DB db = DB.conn();
      db.executeUpdate(querry);
      return newGroup;
    }

    public static Group changeGroup(String oldName, String newName, LocalDate
startDate, LocalDate finishDate,
          List<DayOfWeek> daysOfWeek)
    {
      Group newGroup = new Group(newName, daysOfWeek, startDate, finishDate);
      String querry = "UPDATE Groups SET " + "name = '" + newName + "', daysOfWeek =
'"
            + Group.parseFromDaysOfWeek(daysOfWeek) + "', startDate = '" +
startDate.toString()
            + "', finishDate = '" + finishDate.toString() + "' WHERE name = '" +
oldName + "';";
      DB db = DB.conn();
      db.executeUpdate(querry);
```

```
        return newGroup;
    }

    public static void deleteGroup(String groupName)
    {
        String querry = "DELETE FROM Groups WHERE name = '" + groupName + "';";
        DB db = DB.conn();
        db.executeUpdate(querry);
    }

    public static void addStudent(Student student, String groupName)
    {
        DB db = DB.conn();
        String querry = "INSERT INTO Students_to_groups (studentid, groupid) VALUES ("
+
                                        "(SELECT studentid FROM Students WHERE name =
'" + student.getName() +
                                        "' AND birthdate = '" + student.getBirthday()
+ "' AND ismale = " +
                                        (student.isMale() ? 1 : 0) + " LIMIT 1), " +
                                        "(SELECT groupid FROM Groups WHERE name = '"
+ groupName + "' LIMIT 1));";
        db.executeUpdate(querry);
    }

    public String toString()
    {
        return this.name;
    }
}
```

Java клас Visiting зберігає в собі дані про відвідування певного студента в певній групі в певний день. В ньому зберігається дані про присутність, оцінку та дату заняття. Також в даному класі прописані методи по роботі з даною сутністю, наприклад методи отримання та редагування інформації про відвідування, методи отримання відвідувань певного студента, певної групи тощо (рисунок 1.22).

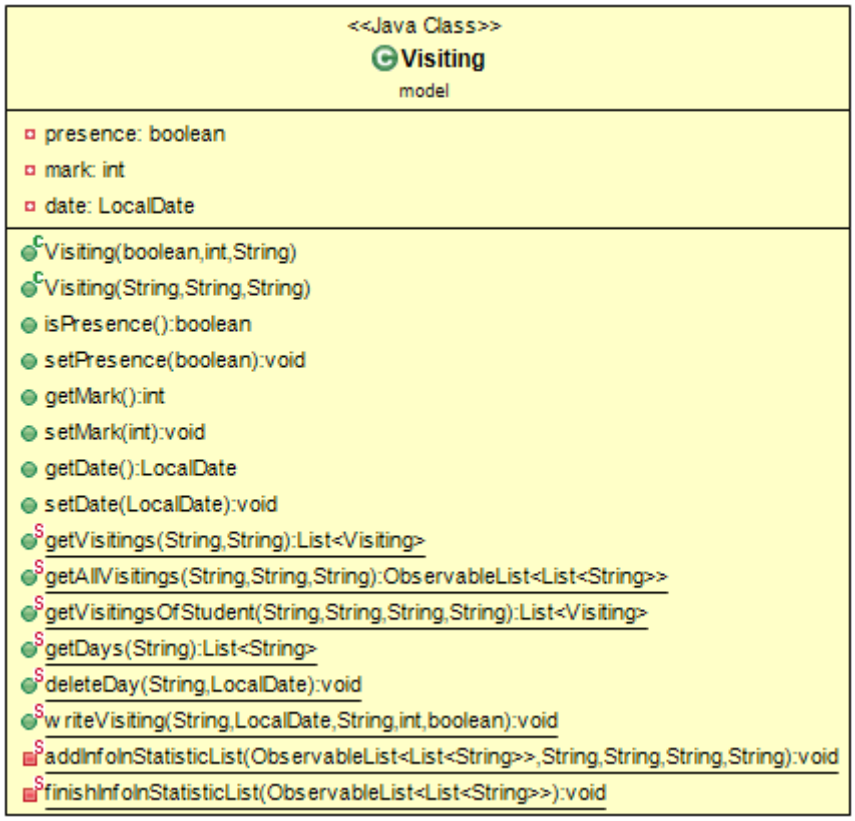


Рисунок 1.22 – Клас Visiting

Основна частина коду класу Visiting зображено в лістингу 1.4.

Лістинг 1.4 – Основна частина класу Visiting

```
package model;

import java.time.LocalDate;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

public class Visiting
{
    private boolean presence;
    private int mark;
    private LocalDate date;

    /**
     *
     * @param presence
     * @param mark
     * @param date
     *             - String date in format "2016-12-31" ("yyyy-mm-dd")
     */
    public Visiting(boolean presence, int mark, String date) {
```

```
        super();
        this.presence = presence;
        this.mark = mark;
        this.date = LocalDate.parse(date);

    }

    public Visiting(String strpresence, String strmark, String strdate) {

        super();
        this.presence = (strpresence == "1");
        if (strmark.equals("null"))
        {
            strmark = "0";
        }
        this.mark = Integer.parseInt(strmark);
        this.date = LocalDate.parse(strdate);
    }

    public boolean isPresence()
    {
        return presence;
    }

    public void setPresence(boolean presence)
    {
        this.presence = presence;
    }

    public int getMark()
    {
        return mark;
    }

    public void setMark(int mark)
    {
        this.mark = mark;
    }

    public LocalDate getDate()
    {
        return date;
    }

    public void setDate(LocalDate date)
    {
        this.date = date;
    }

    public  static  ObservableList<List<String>>  getAllVisitings(String  groupName,
String dateFrom, String dateTo)
    {
        ObservableList<List<String>> mainList = FXCollections.observableArrayList();

        List<Student> listStudents = Student.getStudents(groupName);
        for (int i = 0; i < listStudents.size(); i++)
        {
            List<String> l = new LinkedList<>();
            l.add("" + (i + 1));
            l.add(listStudents.get(i).getName());
            mainList.add(l);
```

```
      }

      String querry = "SELECT student_name, date, mark, presence "
            + "FROM Visitings_of_students WHERE group_name = '"
            + groupName + "' AND ((date >= '" + dateFrom + "' AND date <= '" +
dateTo
            + "') OR date IS NULL) ORDER BY student_name, date;";

      DB db = DB.conn();
      List<String> strList = db.select(querry);

      String curDate = "";
      for (String str : strList)
      {
          String[] split = str.split("\n");
          if (curDate.equals(split[1]) || curDate.equals(""))
          {
             addInfoInStatisticList(mainList,    split[0],    split[1],    split[2],
split[3]);

          } else
          {
             finishInfoInStatisticList(mainList);
             addInfoInStatisticList(mainList,    split[0],    split[1],    split[2],
split[3]);
             curDate = split[1];
          }
      }
      finishInfoInStatisticList(mainList);

      String dates = dateFrom.toString() + " " + dateTo.toString();
      for (List<String> l : mainList)
      {
          List<String>  stat  =  Student.getPresenceStatistic(l.get(1),  groupName,
groupName, dates);
          l.add(stat.get(0));
          l.add(stat.get(1) + "/" + stat.get(2));
      }

      return mainList;
   }

   public  static  List<Visiting>  getVisitingsOfStudent(String  groupName,  String
studentName, String dateFrom,
          String dateTo)
   {
      List<Visiting> visits = new LinkedList<>();
      String querry = "SELECT presence, mark, date "
            + "FROM Visitings_of_students WHERE student_name = '" + studentName
            + "' AND group_name = '" + groupName
            + "' AND (date >= '" + dateFrom + "' AND date <= '"
            + dateTo + "') ORDER BY student_name, date;";

      DB db = DB.conn();
      List<String> strList = db.select(querry);

      for (String str : strList)
      {

          String[] split = str.split("\n");
          visits.add(new Visiting(split[0], split[1], split[2]));
```

```
        }

        return visits;
    }

    /**
     * Method for getting list of days, recorded in DB of some group. Like, in what
     * days group has visits in DB.
     *
     * @param groupName
     *             - name of group
     * @return string list of dates of group, like {"2018-03-10", "2018-03-30"}
     */
    public static List<String> getDays(String groupName)
    {
        List<String> list = new LinkedList<>();

        String querry = "SELECT student_name, date, mark, presence FROM
Visitings_of_students "
              + " WHERE g.name = '" + groupName
              + "' ORDER BY v.date;";

        DB db = DB.conn();
        List<String> dbList = db.select(querry);

        for (String str : dbList)
        {
            String strDate = str.split("\n")[0];
            list.add(strDate);
        }

        return list;
    }

    /**
     * Delete data about visits from BD of chosen day of chosen group
     *
     * @param groupName
     *             name of Group, which to delete
     * @param day
     *             which data about to delete
     */
    public static void deleteDay(String groupName, LocalDate day)
    {
        String querry = "DELETE FROM Visitings WHERE date = '" + day.toString()
              + "' AND groupid IN (SELECT groupid FROM Groups WHERE name = '" +
groupName + "');";

        DB db = DB.conn();
        db.executeUpdate(querry);
    }

    /**
     * Write info about visiting in DB
     *
     * @param groupName
     *             - name of group
     * @param day
     *             - day of visiting to write
     * @param studentName
     *             - name of student
```

```
     * @param mark
     *            - mark, what student got
     * @param presence
     *            - was student presence
     */
    public static void writeVisiting(String groupName, LocalDate date, String
studentName, int mark, boolean presence)
    {

       String querry = "INSERT INTO Visitings (groupid, studentid, mark, presence,
date) VALUES ("
             + "(SELECT groupid FROM Groups WHERE name = '" + groupName + "'), "
             + "(SELECT studentid FROM Students WHERE name = '" + studentName
             + "' AND studentid IN (SELECT studentid FROM Students_to_groups"
             + " WHERE groupid IN (SELECT groupid FROM Groups WHERE name = '" +
groupName + "'))), "
             + (mark == 0 ? "null" : mark) + ", " + (presence ? 1 : 0) + ", '" +
date.toString() + "');";
       DB db = DB.conn();
       db.executeUpdate(querry);
    }

    private static void addInfoInStatisticList(ObservableList<List<String>> mainList,
String name, String date,
          String mark, String presence)
    {
       for (List<String> l : mainList)
       {
          if (l.get(1).equals(name))
          {
             l.add(date);
             l.add(mark);
             l.add(presence);
          }
       }
    }

    private   static   void   finishInfoInStatisticList(ObservableList<List<String>>
mainList)
    {
       int maxSize = 2;
       List<String> maxList = new LinkedList<>();
       for (List<String> l : mainList)
       {
          if (l.size() > maxSize)
          {
             maxSize = l.size();
             maxList = l;
          }
       }
       for (int i = 0; i < maxSize; i++)
       {
          for (List<String> l : mainList)
          {
             while (l.size() < maxSize)
             {
                l.add(maxList.get(l.size() - 1));
                l.add("null");
                l.add("null");
             }
          }
```

```
        }
    }
}
```

Клас DB створений для роботи з базою даних. Практично всі взаємодії програми з БД проводяться через об’єкт класу DB. Даний клас містить в собі реалізований шаблон проектування Singleton [22]. Також в ньому реалізовані методи для створення нового файлу БД, виконання SQL запитів для отримання та редагування даних БД, тощо (рисунок 1.23).

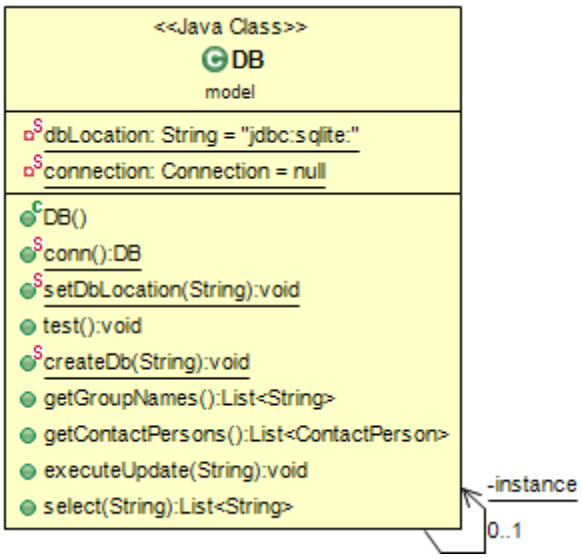


Рисунок 1.23 – Клас DB

Основні частини коду класу DB

Лістинг 1.5 – Основні частини класу DB

```
package model;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.LinkedList;
import java.util.List;

public class DB
{
    private static String dbLocation = "jdbc:sqlite:";
    private static DB instance;
    private static Connection connection = null;

    public static DB conn()
    {
```

```
   try
   {
       if (connection == null)
       {
          connection = DriverManager.getConnection(dbLocation);
       }
       if (instance == null)
       {
          instance = new DB();
       }
   } catch (SQLException e)
   {
       e.printStackTrace();
   }
   return instance;
}

public static void setDbLocation(String urlToDb)
{
   dbLocation = "jdbc:sqlite:" + urlToDb;
   System.out.println("New db location: " + dbLocation);
}

public void test()
{
   try
   {
       // create a database connection
       connection = DriverManager.getConnection("jdbc:sqlite:sample.db");
       Statement statement = connection.createStatement();
       statement.setQueryTimeout(30); // set timeout to 30 sec.

       statement.executeUpdate("drop table if exists person");
       statement.executeUpdate("create table person (id integer, name string)");
       statement.executeUpdate("insert into person values(1, 'leo')");
       statement.executeUpdate("insert into person values(2, 'yui')");
       ResultSet rs = statement.executeQuery("select * from person");
       while (rs.next())
       {
          // read the result set
          System.out.println("name = " + rs.getString("name"));
          System.out.println("id = " + rs.getInt("id"));
       }
   } catch (SQLException e)
   {
       // if the error message is "out of memory",
       // it probably means no database file is found
       System.err.println(e.getMessage());
   } finally
   {
       try
       {
          if (connection != null)
              connection.close();
       } catch (SQLException e)
       {
          // connection close failed.
          System.err.println(e);
       }
   }
}
```

```
    public static void createDb(String filePath)
    {
       String oldDbLocation = dbLocation;
       setDbLocation(filePath);

       DB db = conn();

       String createTableQuarry = "PRAGMA foreign_keys = off;\r\n" +
                   "BEGIN TRANSACTION;\r\n" +
                   "\r\n" +
                   "-- Table: ContactPersons\r\n" +
                   "DROP TABLE IF EXISTS ContactPersons;\r\n" +
                   "CREATE TABLE ContactPersons (contactpersonid INTEGER PRIMARY KEY
AUTOINCREMENT UNIQUE NOT NULL, name TEXT NOT NULL, address TEXT NOT NULL, phone TEXT
NOT NULL);\r\n" +
                   "\r\n" +
                   "-- Table: Groups\r\n" +
                   "DROP TABLE IF EXISTS Groups;\r\n" +
                   "CREATE TABLE \"Groups\" (\r\n" +
                   "      `groupid`    INTEGER  NOT  NULL  PRIMARY  KEY  AUTOINCREMENT
UNIQUE,\r\n" +
                   "      `name` TEXT NOT NULL,\r\n" +
                   "      `daysOfWeek` TEXT,\r\n" +
                   "      `startDate`  TEXT NOT NULL,\r\n" +
                   "      `finishDate` TEXT NOT NULL\r\n" +
                   ");\r\n" +
                   "\r\n" +
                   "-- Table: Students\r\n" +
                   "DROP TABLE IF EXISTS Students;\r\n" +
                   "CREATE  TABLE  Students  (studentid  INTEGER  NOT  NULL  PRIMARY  KEY
AUTOINCREMENT UNIQUE, groupid INTEGER, name TEXT NOT NULL, birthdate TEXT NOT NULL,
ismale INTEGER NOT NULL DEFAULT 1, contactpersonid INTEGER REFERENCES ContactPersons
(contactpersonid)  ON  DELETE  SET  NULL,  FOREIGN  KEY  (groupid)  REFERENCES  Groups
(groupid) ON DELETE CASCADE);\r\n" +
                   "\r\n" +
                   "-- Table: Students_to_groups\r\n" +
                   "DROP TABLE IF EXISTS Students_to_groups;\r\n" +
                   "CREATE  TABLE  Students_to_groups  (students_to_groupsid  INTEGER
PRIMARY KEY AUTOINCREMENT UNIQUE NOT NULL, studentid INTEGER REFERENCES Students
(studentid) ON DELETE CASCADE NOT NULL, groupid INTEGER REFERENCES Groups (groupid)
ON DELETE CASCADE NOT NULL);\r\n" +
                   "\r\n" +
                   "-- Table: Visitings\r\n" +
                   "DROP TABLE IF EXISTS Visitings;\r\n" +
                   "CREATE TABLE \"Visitings\" (\r\n" +
                   "      `visitingid` INTEGER  NOT  NULL  PRIMARY  KEY  AUTOINCREMENT
UNIQUE,\r\n" +
                   "      `groupid`    INTEGER NOT NULL,\r\n" +
                   "      `studentid`  INTEGER NOT NULL DEFAULT 1,\r\n" +
                   "      `presence`   INTEGER NOT NULL DEFAULT 0,\r\n" +
                   "      `mark` INTEGER,\r\n" +
                   "      `date` TEXT NOT NULL,\r\n" +
                   "      FOREIGN  KEY(`groupid`)  REFERENCES  `Groups`(`groupid`)  ON
DELETE cascade,\r\n" +
                   "      FOREIGN             KEY(`studentid`)             REFERENCES
`Students`(`studentid`) ON DELETE cascade\r\n" +
                   ");\r\n" +
                   "\r\n" +
                   "-- View: Contactpersons_of_students\r\n" +
                   "DROP VIEW IF EXISTS Contactpersons_of_students;\r\n" +
```

```
                    "CREATE VIEW Contactpersons_of_students AS SELECT s.name AS
student_name, s.birthdate, s.ismale, c.name AS contactperson_name, c.address,
c.phone\r\n" +
                    "FROM ContactPersons AS c INNER JOIN Students AS s ON
s.contactpersonid = c.contactpersonid;\r\n" +
                    "\r\n" +
                    "-- View: Students_in_groups\r\n" +
                    "DROP VIEW IF EXISTS Students_in_groups;\r\n" +
                    "CREATE VIEW Students_in_groups AS SELECT s.name AS student_name,
s.birthdate, s.ismale, g.name AS group_name\r\n" +
                    "FROM Students AS s INNER JOIN Students_to_groups AS sg ON
s.studentid = sg.studentid INNER JOIN Groups AS g ON g.groupid = sg.groupid;\r\n" +
                    "\r\n" +
                    "-- View: Visitings_of_students\r\n" +
                    "DROP VIEW IF EXISTS Visitings_of_students;\r\n" +
                    "CREATE VIEW Visitings_of_students AS SELECT s.name AS
student_name, s.birthdate, s.ismale, g.name AS group_name, v.date, v.mark, v.presence
FROM Students AS s INNER JOIN Students_to_groups AS stg ON s.studentid =
stg.studentid\r\n" +
                    "INNER JOIN Groups AS g ON stg.groupid = g.groupid INNER JOIN
Visitings AS v ON v.studentid = s.studentid AND v.groupid = g.groupid;\r\n" +
                    "\r\n" +
                    "COMMIT TRANSACTION;\r\n" +
                    "PRAGMA foreign_keys = on;\r\n" +
                    "";
      db.executeUpdate(createTableQuarry);
      setDbLocation(oldDbLocation.substring(12, oldDbLocation.length()));
   }

   public List<ContactPerson> getContactPersons()
   {
      List<ContactPerson> contactList = new LinkedList<ContactPerson>();
      try
      {
          // create a database connection
          connection = DriverManager.getConnection(dbLocation);
          Statement statement = connection.createStatement();
          statement.setQueryTimeout(30); // set timeout to 30 sec.

          ResultSet rs = statement.executeQuery("SELECT name, address, phone FROM
ContactPersons");
          while (rs.next())
          {
             contactList.add(new ContactPerson(rs.getString("name"),

      rs.getString("address"),

      rs.getString("phone")));
          }
      } catch (SQLException e)
      {
          System.err.println(e.getMessage());
      } finally
      {
          try
          {
             if (connection != null)
                 connection.close();
          } catch (SQLException e)
          {
             System.err.println(e);
```

```
        }
    }
    return contactList;
}

public void executeUpdate(String querry)
{
    System.out.println(querry);
    // Connection connection = null;
    try
    {
        // create a database connection
        connection = DriverManager.getConnection(dbLocation);
        Statement statement = connection.createStatement();
        statement.setQueryTimeout(30); // set timeout to 30 sec.
        statement.executeUpdate(querry);
    } catch (SQLException e)
    {
        // if the error message is "out of memory",
        // it probably means no database file is found
        System.err.println(e.getMessage());
    } finally
    {
        try
        {
          if (connection != null)
              connection.close();
        } catch (SQLException e)
        {
          // connection close failed.
          System.err.println(e);
        }
    }
}

public List<String> select(String querry)
{
    System.out.println(querry);
    // Connection connection = null;
    List<String> list = new LinkedList<String>();
    try
    {
        connection = DriverManager.getConnection(dbLocation);
        Statement statement = connection.createStatement();
        statement.setQueryTimeout(30); // set timeout to 30 sec.

        ResultSet rs = statement.executeQuery(querry);
        ResultSetMetaData rsmd = rs.getMetaData();

        int columnsNumber = rsmd.getColumnCount();
        String current = "";
        while (rs.next())
        {
          current = "";
          int i = 1;
          while (i <= columnsNumber)
          {
              current += rs.getString(i++) + "\n";
          }
          list.add(current);
        }
```

```
        } catch (SQLException e)
        {
            System.err.println(e.getMessage());
        } finally
        {
            try
            {
               if (connection != null)
                   connection.close();
            } catch (SQLException e)
            {
               System.err.println(e);
            }
        }

        /*
         * System.out.println("data from SELECT:"); for(String s : list) {
         * System.out.println("\"" + s + "\""); }
         */

        return list;
    }

}
```

Клас ContactPerson призначений для зберігання інформації про контактну особу учня/студента. Зберігаються такі дані, як ім'я, адреса проживання та контактний телефон. В класі реалізовані методи для отримання та редагування цих атрибутів. Також є методи для прив'язки та відв'язки контактної особи для учня/студента (рис. 1.24)

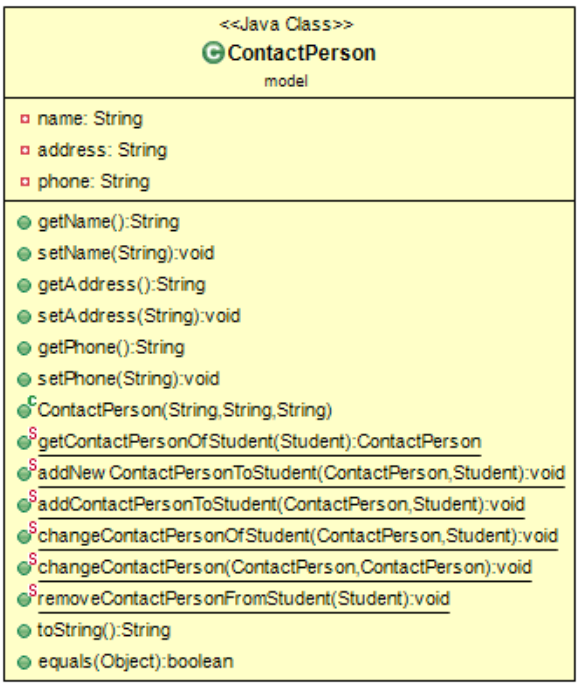
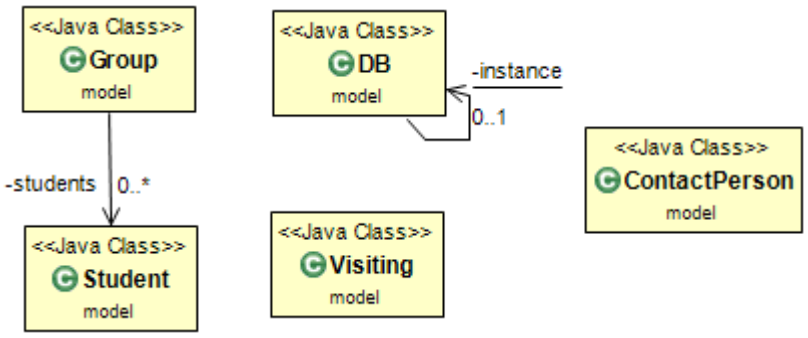


Рис. 1.24 – Клас ContactPerson

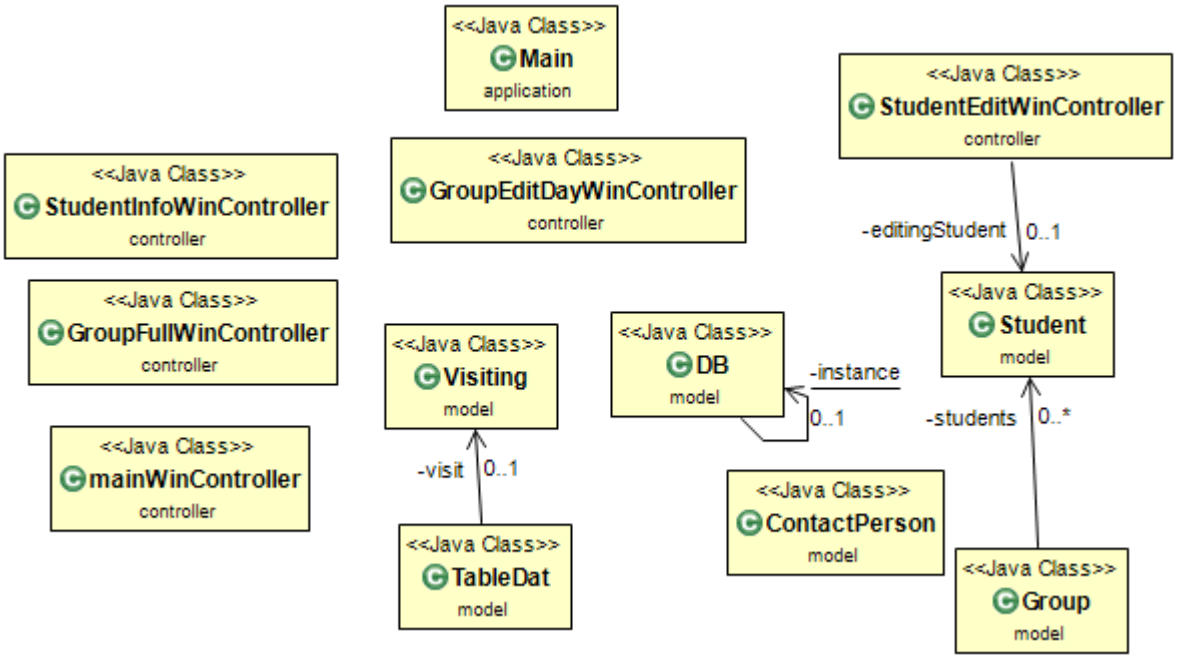
Скомпонувавши дані класи, отримаємо діаграму зв’язків між згаданими класами на рисунку 1.25.



1.25 – Зв’язки основних класів моделі програми

Детальнішу діаграму класів моделі програми зображено в додатку Г.

Скомпонувавши всі основні класи програми, отримаємо загальнішу діаграму класів програми (рис. 1.26).



1.26 – Діаграма класів

Проаналізувавши написаний код, можна дійти висновку, що програма в цілому написана за основними критеріями ООП. В програмі використовуються коментарі та досить ясні назви назв, полей та методів класів і змінних, що підвищує зрозумілість та читабельність коду в цілому.

1.4 Використання програмної системи

1.4.1 Розгортання програмної системи та системні вимоги

Програма складається з трьох компонентів:

а) Середовища запуску Джава 8 версії (Java Virtual machine, JVM).

б) Власне, програма (файл Program.jar).

в) База даних (БД, файл students.db).

Щоб перевірити, чи на комп’ютері є JVM необхідної версії, необхідно скачати файл програми з сторінки проекту на сайті GitHub та спробувати її відкрити. Якщо комп’ютер не зміг відкрити програму, отже необхідно встановити JVM.

Для цього, слід перейти за посиланням : https://java.com/ru/download/ , натиснути кнопку «Загрузить Java бесплатно»,прочитати та погодитися з умовами використання JVM, натиснути «Согласиться и начать бесплатную загрузку». Після цього почнеться завантаження встановлювача JVM для операційної системи та розрядності Вашого ком’ютера (наприклад Windows, 64-bit чи MacOs, 32-bit, це вибирається автоматично). Після скачення встановлювача запустити його та провести встановлення, слідуючи інтструкції. Після встановлення рекомендовано перезавантажити комп'ютер.

Системні вимоги для використання Java 8 такі:

Windows

- Windows 10 (8u51 і вище)
- Windows 8.x (настільний ПК)
- Windows 7 SP1
- Windows Vista SP2
- Windows Server 2008 R2 SP1 (64-розрядна)
- Windows Server 2012 та 2012 R2 (64-розрядні)
- ОЗУ: 128 Мб
- Місце на диску: 124 Мб для JRE; 2 Мб для оновлення Java

– Процесор: мінімальний процесор Pentium 2 266 МГц

– Браузери: Internet Explorer 9 і новіших версій, Firefox

Mac OS X

– Mac на базі Intel, що працює на Mac OS X 10.8.3+, 10.9+

– Права адміністратора для встановлення

– 64-розрядний браузер

Для запуску Oracle Java на Mac потрібен 64-розрядний браузер (наприклад, Safari).

Linux

- Oracle Linux 5.5+ 1
- Oracle Linux 6.x (32-бітний), 6.x (64-розрядний) 2
- Oracle Linux 7.x (64-розрядні) 2 (8u20 і вище)
- Red Hat Enterprise Linux 5.5+ 1 , 6.x (32-бітний), 6.x (64-бітний) 2
- Red Hat Enterprise Linux 7.x (64-розрядні) 2 (8u20 і вище)
- Suse Linux Enterprise Server 10 SP2 +, 11.x
- Suse Linux Enterprise Server 12.x (64-бітний) 2 (8u31 і вище)
- Ubuntu Linux 12.04 LTS, 13.x
- Ubuntu Linux 14.x (8u25 і вище)
- Ubuntu Linux 15.04 (8u45 і вище)
- Ubuntu Linux 15.10 (8u65 і вище)
- Браузери: Firefox [23]

Детальніше про перевірку наявності та встановлення JVM можна переглянути в спеціалізованих підручниках по використанню JVM [24].

Для скачування програми, необхідно перейти на сторінку програми на сайті GitHub та скачати файл Program.jar. Для цього спочатку натиснути на «Program.jar» (рисунок 1.24).

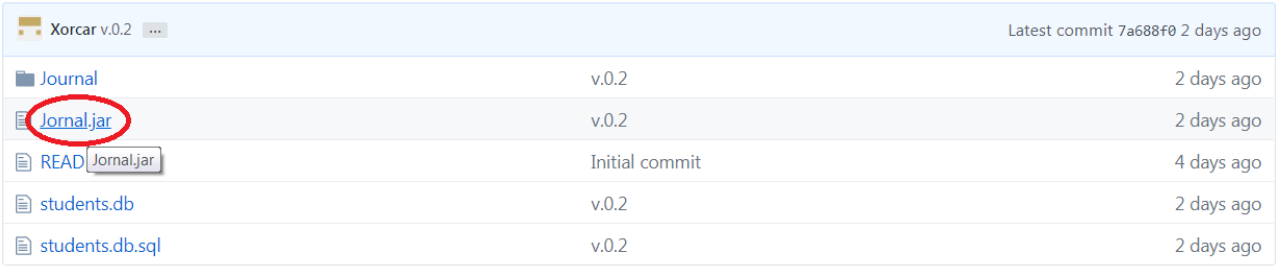


Рисунок 1.24 – Вибір програми для її скачування

І потім на кнопку «Download» (рисунок 1.25).

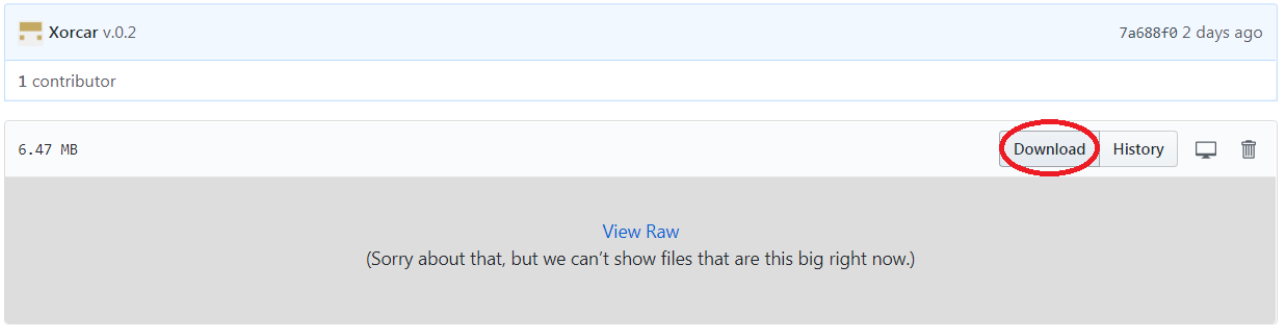


Рисунок 1.25 – Скачування програми

Встановлювати саму програму НЕ необхідно, вона вже готова до роботи.

Для скачування бази даних, слід перейти на сторінку проекту на сайті GitHub та скачати файл students.db . Після скачування, її необхідно помістити в теку з файлом Program.jar для коректної роботи.

Файл students.db – це тестова база даних, в якій вже є записані тестові дані для ознайомлення з програмою. Після ознайомлення, рекомендуємо файл students.db видалити з комп'ютера та замінити її на файл бази даних без записів. Це можна зробити двома способами:

а) Скачати з сторінки проекту файл studentsRaw.db, перемістити його в теку з програмою та перейменувати в students.db.

б) Або створити порожній файл бази даних за допомогою самої програми.

Після цього програма повністю встановлена і готова до використання.

1.4.2 Верифікація програмної системи

Для верифікація було побудовано таблиці з вимогами та перевірено, чи відповідає розроблена програма даним вимогам (таблиця 1.13).

Таблиця 1.13 – Верифікація програмної системи

Вимога	Чи реалізовано
Мультиплатформеність	так
Простота інтерфейсу	так
Можливість використовувати на різних комп’ютерах	так
ВВ В1 «Заповнення журналу проведеного уроку»	так
ВВ В2 «Перегляд журналу за попередні дні»	так
ВВ В3 «Перегляд статистики учня»	так
ВВ В4 «Зміна даних журналу за певне попереднє заняття»	так
ВВ В5 Створення групи	так
ВВ В6 Додавання студентів до групи	так
ВВ В7 Видалення студентів з групи	так
ВВ В8 Редагування даних студента	так
ВВ В9 Редагування даних групи	так
Облік відвідування та успішності учнів	так

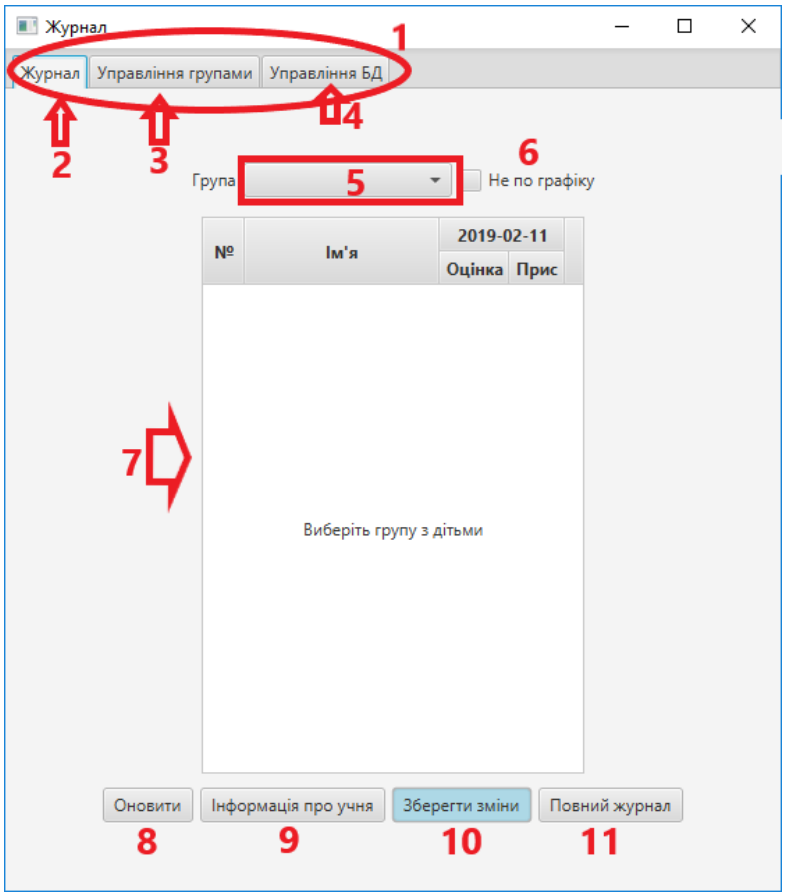
Провевши верифікацію програмної системи, можна дійти висновку, що розроблена система повністю відповідає поставленим до неї вимогам.

2. ІНСТРУКЦІЯ ВИКОРИСТАННЯ ПРОГРАМНОГО ПРОДУКТУ

Як тестування розробленого продукту, було проведена ручна перевірка функцій додатку. Результат перевірки у вигляді інструкції використання додатку наведено в цьому розділі.

2.1 Загальна інструкція користування додатком

Після запуску програми, буде зображено першу закладку головного вікна програми (рисунок 2.1).



1 – закладки програми;
2 – закладка «Журнал»;
3 – закладка «Управління групами»;
4 – закладка «Управління БД»;
5 – випадаючий список вибору групи;
6 – прапорець проведення заняття не по графіку;
7 – таблиця сьогоднішнього журналу;
8 – кнопка оновлення таблиці;
9 – виклик вікна інформації про вибраного в таблиці учня;
10 – зберігання введених оцінок та присутностей в базу даних
11 – виклик повного журналу вибраної групи.

Рисунок 2.1 – Закладка журналу головного вікна

Заповнення оцінок та присутностей на поточний день.

У випадаючому списку вибору групи знаходиться список груп, заняття яких заплановано на сьогодні (ті групи, навчальний період яких вже

розпочався, ще не закінчився і заняття проводиться в поточний день тижня). При виборі прапорця «Не по графіку» (рисунок 2.1, №6) даний список буде заповнений всіма групами з бази даних (БД). Після вибору групи з випадаючого списку, таблиця №7 заповниться (рисунок 2.2)

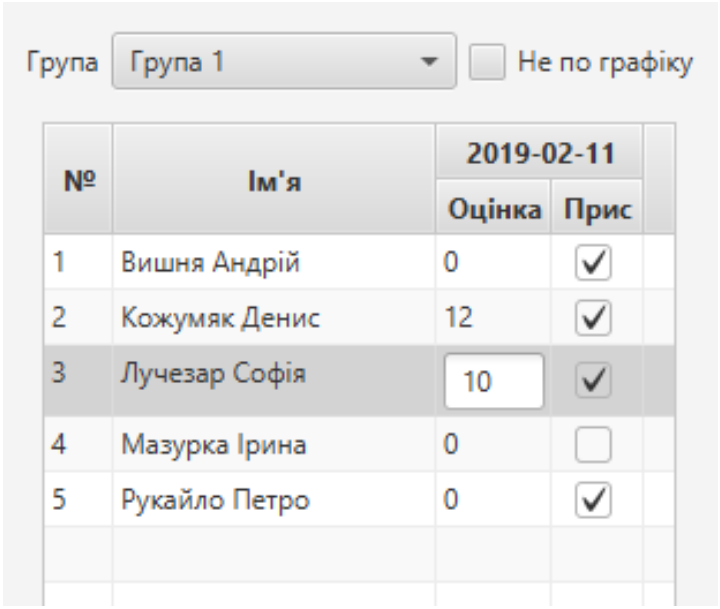


Рисунок 2.2 – Журнал на 2019-02-11 для групи Група 1

Таблиця на рисунку 2.2 складається з колонки номера п/п, колонки ім’я та колонки сьогоднішньої дати. Колонка дати складається з колонок оцінки та присутності. Якщо у БД є запис сьогоднішнього дня для вибраної групи, таблиця заповняться оцінками та при сутностями з БД. Якщо ж у БД немає записів на сьогодні про вибрану групу, таблиця заповниться списком дітей і всім дітям буде виставлена оцінка 0 та відсутність в таблиці.

Для відмічення присутності студента, необхідно поставити відмітку в квадратик стовпця «Прис» навпроти імені учня. Для виставленні оцінки, необхідно в стовпці «Оцінка» вписати оцінку та натиснути Enter для підтвердження вводу.

На рисунку 2.2 вказані присутність учнів Вишня Андрій, Кожумяк Денис, Лучезар Софія та Рукайло Петро. У учня Кожумяк Денис виставлена оцінка 12. У Лучезар Софії вказана (але не підтверджена) оцінка 10. У решти учнів підтверджена оцінка 0. Вибраний учень – Лучезар Софія.

Після виставлення присутностей та оцінок і підтвердження оцінок, можна натискати кнопку «Зберегти зміни» (№10 на рисунку 2.1). Ця кнопка запише дані (оцінки та присутність) учнів групи на сьогоднішній день в БД.

Перегляд журналу групи.

Після вибору групи в випадаючому списку (№5 на рисунку 2.1) на натискання кнопки «Повний журнал», викличеться вікно повного журналу групи (рисунок 2.3)

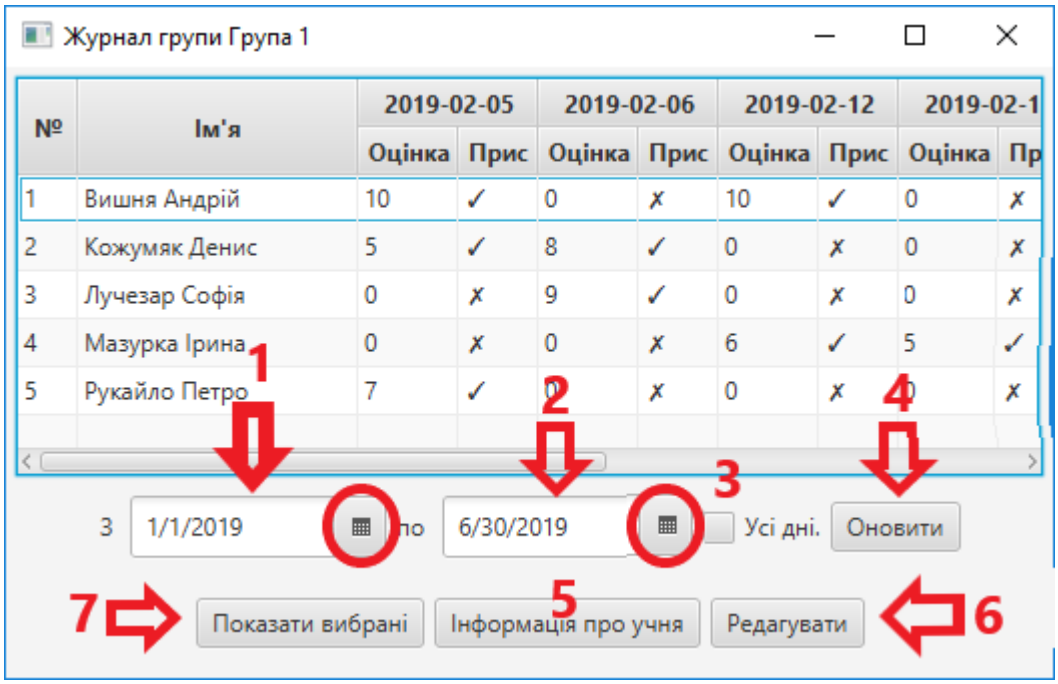


Рисунок 2.3 – Вікно повного журналу групи Група 1

1 – Дата початку відліку. 2 – Дата кінця відліку. 3 – Прапорець перегляду журналу по усіх днях. 4 – Кнопка оновлення журналу. 5 – Кнопка інформації вибраного в таблиці учня. 6 – Кнопка редагування журналу. 7 – Кнопка для перемикання між показом усіх учнів групи та між показом тільки вибраних користувачем.

Вікно повного журналу показує усі оцінки та присутності за вибраний період у попередньо вибраній групі. В кінці таблиці вказано середній бал (нулі не враховуються та кількість присутностей/відсутностей; якщо учень за вибраний період не має оцінок (усі нулі), то показується «null») За замовчуванням, показується історія за поточне півріччя.

Для зміни проміжку часу, за який слід показувати журнал, можна змінити дати початку та кінця відліку журналу (обведені червоним на рисунку 2.3).

Для показу журналу групи повністю (усі записи групи) слід дів значити (натиснути на) прапорець «Усі дні» (№3 на рисунку 2.3).

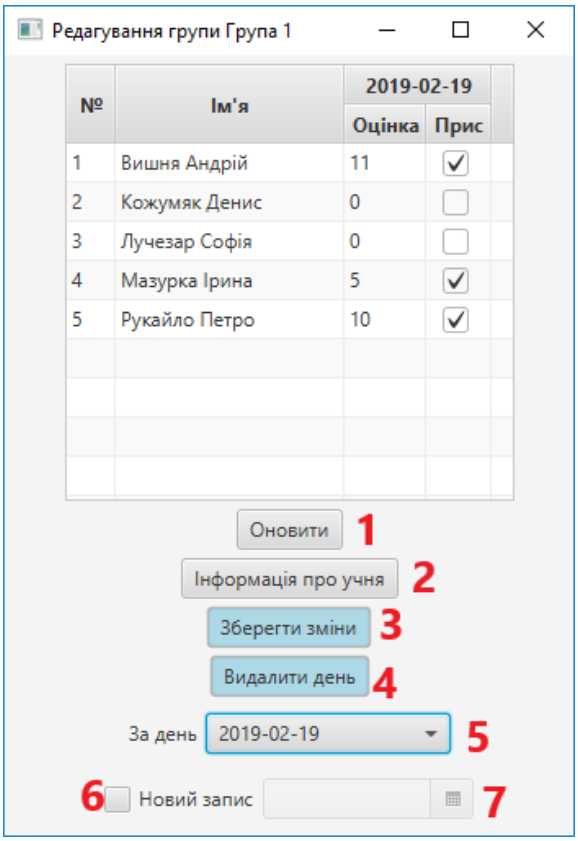
Для оновлення таблиці, слід натиснути на кнопку «Оновити» (№4 на рисунку 2.3).

Для перегляду інформації про учня, слід обрати його в таблиці та натиснути кнопку «Інформація про учня» (№5 на рисунку 2.3).

Для редагування записів відвідуваності групи або додавання записів за минулі дні, слід натиснути кнопку «Редагувати» (№6, рисунку 2.3).

Редагування журналу

Якщо у вікні повного журналу групи натиснути кнопку «Редагувати» (№6 на рисунку 2.3), викличеться вікно редагування журналу (рисунок 2.4).



1 – Кнопка оновлення таблиці.

2 – Кнопка перегляду інформації про учня.

3 – Кнопка збереження змін.

4 – Кнопка видалення дня.

5 – Випадаючий список днів для редагування.

6 – Прапорець створення нового запису.

7 – Дата нового запису.

Рисунок 2.4 – Вікно редагування журналу

При відкритті вікна, таблиця буде без записів та з проханням «Виберіть день». Стовбець дати матиме запис «0000-00-00». Після вибору дати через вибирач дат (№7), таблиця заповниться списком учнів з розставленими нулями як оцінками та відсутностями. При виборі дати з випадаючого списку (№5), таблиця заповниться оцінками та відсутностями за вибраний день з БД.

Правила заповнення таблиці аналогічні до правил, написаних попередньо.

Для оновлення таблиці, слід натиснути кнопку «Оновити» (№1).

Для перегляду інформації про учня, його слід обрати в таблиці та натиснути кнопку «Інформація про учня» (№2).

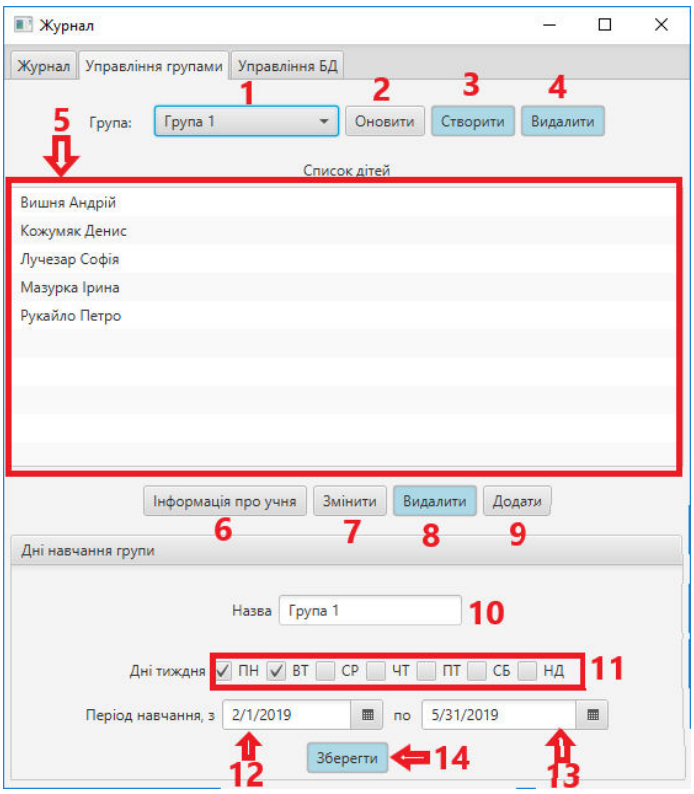
Для збереження записів, введених в таблицю, в БД, слід натиснути кнопку «Зберегти зміни» (№3).

Для вибору дня для редагування, слід скористатися випадаючим списком №5.

Для вибору нового дня для запису в журнал (наприклад, якщо слід ввести дані за минулий день (за вчора)), слід відзначити прапорець «Новий запис» (№6) та обрати дату через вибирач дат №7 (натиснувши його значок календарика та вибравши бажану дату).

2.2. Інструкція управління групами та студентами

Для управління групами і студентами, слід перейти на вкладку «Управління групами» головного вікна програми (№3, рисунок 2.1). Вікно перейде на закладку управління групами (рисунок 2.5)



1 – Випадаючий список груп.

2 – Кнопка оновлення списків груп.

3 – Кнопка створення групи.

4 – Кнопка видалення групи.

5 – Список учнів вибраної групи.

6 – Кнопка інформації про вибраного в списку №5 учня.

7 – Кнопка зміни вибраного в списку №5 учня.

8 – Додавання до вибраної в випадаючому списку №1 учня

9 – Видалення вибраного учня з вибраної групи.

10 – Текстове поле зміни назви вибраної групи.

11 – Вибір днів навчання групи.

12 – Дата початку навчання групи.

13 – Дата закінчення навчання групи.

14 – Кнопка збереження змін, введених в №9 - №12 у вибрану групу.

Рисунок 2.5 – Закладка управління групами головного вікна програми

Управління групами.

Для вибору групи для редагування, її слід обрати в випадаючому списку №1. Після вибору групи, список №5 заповниться учнями цієї групи, а поля №9-№12 заповняться даними про неї.

Для оновлення списку груп №1 та №5, рисунок 2.1, слід натиснути кнопку «Оновити» (№2).

Для створення нової групи, слід натиснути кнопку №3.

Для видалення вибраної групи, слід натиснути кнопку «Видалити» (№4).

Для перегляду інформації про учня, його слід обрати в списку №5 та натиснути кнопку «Інформація про учня» (№6).

Для зміни інформації про учня, слід вибрати його в списку №5 та натиснути кнопку «Змінити» (№7).

Для видалення учня з групи, слід вибрати його в списку групи та натиснути кнопку «Видалити» (№8). Видалення учня з групи видалить всі його записи в даній групі, але не зачепить дані цього ж учня в інших групах.

Для додавання учня до вибраної групи, слід натиснути кнопку «Додати» (№9).

Для зміни назви групи, її слід передрукувати в текстове поле №10.

Для зміни робочих днів групи, необхідно вибрати бажані дні тижня та зняти виділення з небажаних в області №11.

Для зміни періоду навчання групи, слід змінити дату початку та закінчення навчання через вибирачі дат №12 та №13 відповідно.

Для затвердження змін, введених в полях №10-№13, слід натиснути кнопку «Зберегти» (№14). ВАЖЛИВО: при збережені змін групи, впевніться, що група вибрана в випадаючому списку №1.

Додавання групи.

Після натискання кнопки «Створити» №3, рисунок 2.5, викличеться вікно створення нової групи (рисунок 2.6)

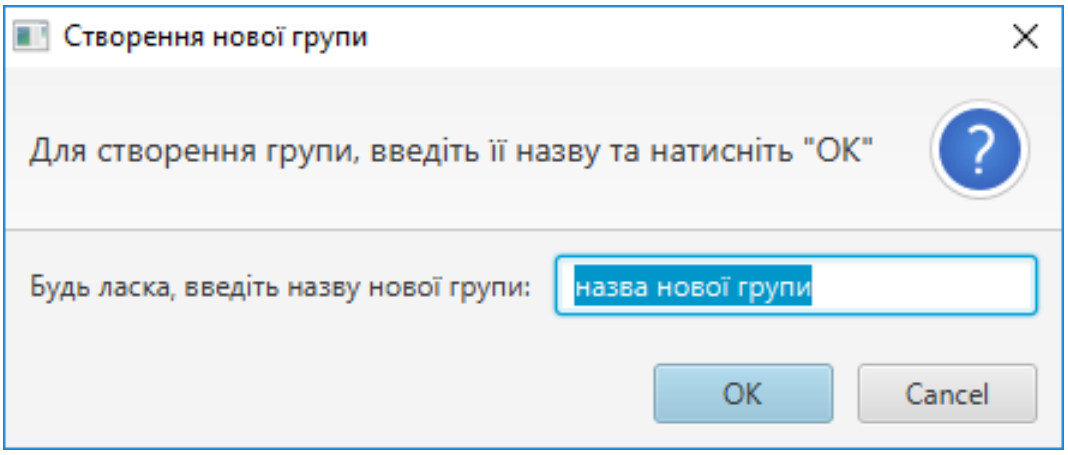


Рисунок 2.6 – Вікно створення нової групи

Після введення назви нової групи в текстове поле та натисканні кнопки «ОК», в базу даних запишеться нова група з вказаною назвою, без учнів, яка вчить ні по яких днях тижня та початок і кінець навчання якої – поточна дата.

ВАЖЛИВО: після створення групи, необхідно оновити випадаючі списки груп №5, рисунок 2.1 та №1, рисунок 2.5, натиснувши на кнопку «Оновити», №2, рисунок 2.5.

Для відміни операції додавання групи, натисніть кнопку «Cancel» або на червону кнопку в верхньому кутку вікна.

Видалення групи.

Після вибору групи в випадаючому списку №1, рисунок 2.5 та натискання кнопки «Видалити», №4, рисунок 2.5, викличеться вікно підтвердження видалення групи (рисунок 2.7).

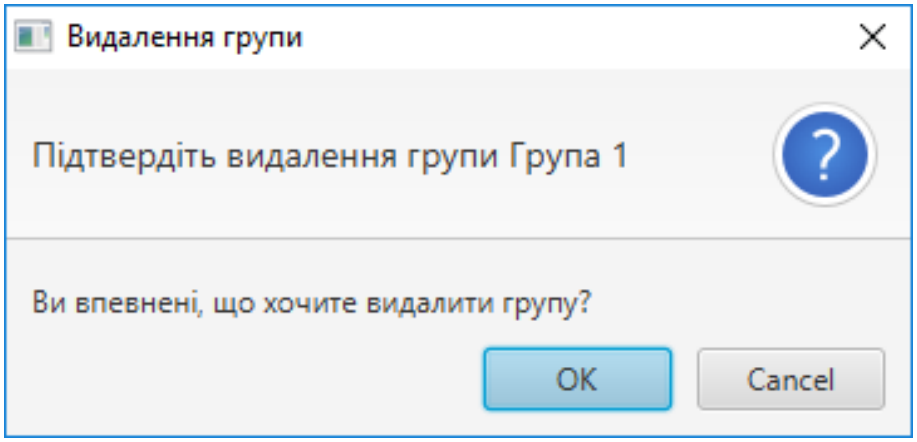


Рисунок 2.7 – Вікно підтвердження видалення групи

Для видалення групи з БД, натисніть кнопку «ОК». Ця дія видалить саму групу, її учнів та журнал усіх днів з БД. Ця дія не зачепить дані інших груп. Ця дія не зачепить дані учнів в інших групах, навіть якщо ці учні є в видаленій групі.

Для відміни операції видалення групи, натисніть кнопку «Cancel» або на червону кнопку в верхньому кутку вікна.

Видалення учня з групи

Для видалення учня з групи, виберіть групу з випадаючого списку №1, рисунок 2.5 потрібну групу та виберіть потрібного учня з списку №5, рисунок 2.5. Після цього, натисніть кнопку «Видалити» (№8, рисунок 2.5).

Це викличе вікно підтвердження видалення учня (рисунок 2.8).

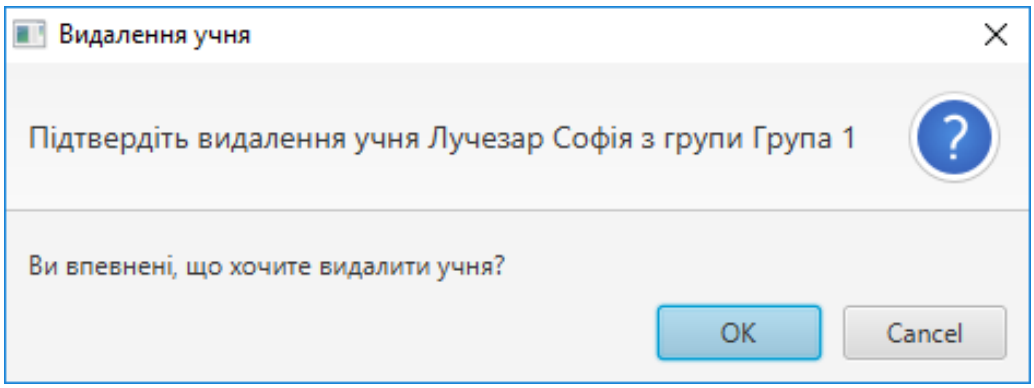


Рисунок 2.8 – Вікно підтвердження видалення учня з групи

Для видалення учня з групи у БД, натисніть кнопку «ОК». Ця дія видалить учня тільки в вибраній групі та його дані про відвідування в вибраній групі. Його дані в інших групах не будуть зачеплені.

Для відміни операції видалення учня з групи, натисніть кнопку «Cancel» або на червону кнопку в верхньому кутку вікна.

Додавання учня до групи

Для додавання учня до групи, виберіть групу в випадаючому списку №1, рисунок 2.5 та натисніть кнопку «Додати» (№9, рисунок 2.5). Після цього викличеться вікно додавання нового учня (рисунок 2.9)

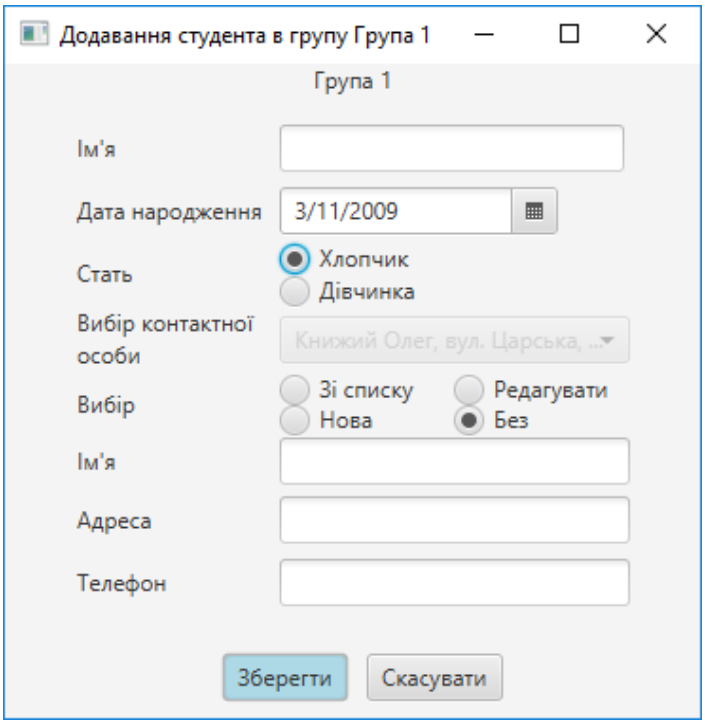


Рисунок 2.9– Вікно додавання учня до групи.

Після введення імені в текстове поле, дня народження в вибирач дати, вибору статі, вибору контактної особи та натисканні кнопки «Зберегти», учень з вказаними характеристиками (ім’я, дата народження, стать, контактна особа) буде записаний у вказану групу.

Для відміни операції додавання учня в групу, натисніть кнопку «Скасувати» або на червону кнопку в верхньому кутку вікна.

Редагування учня до групи

Для додавання учня до групи, виберіть групу в випадаючому списку №1, рисунок 2.5, необхідного учня в списку №5, рисунок 2.5 та натисніть кнопку «Змінити» (№7, рисунок 2.5). Після цього викличеться вікно редагування нового учня (рисунок 2.10)

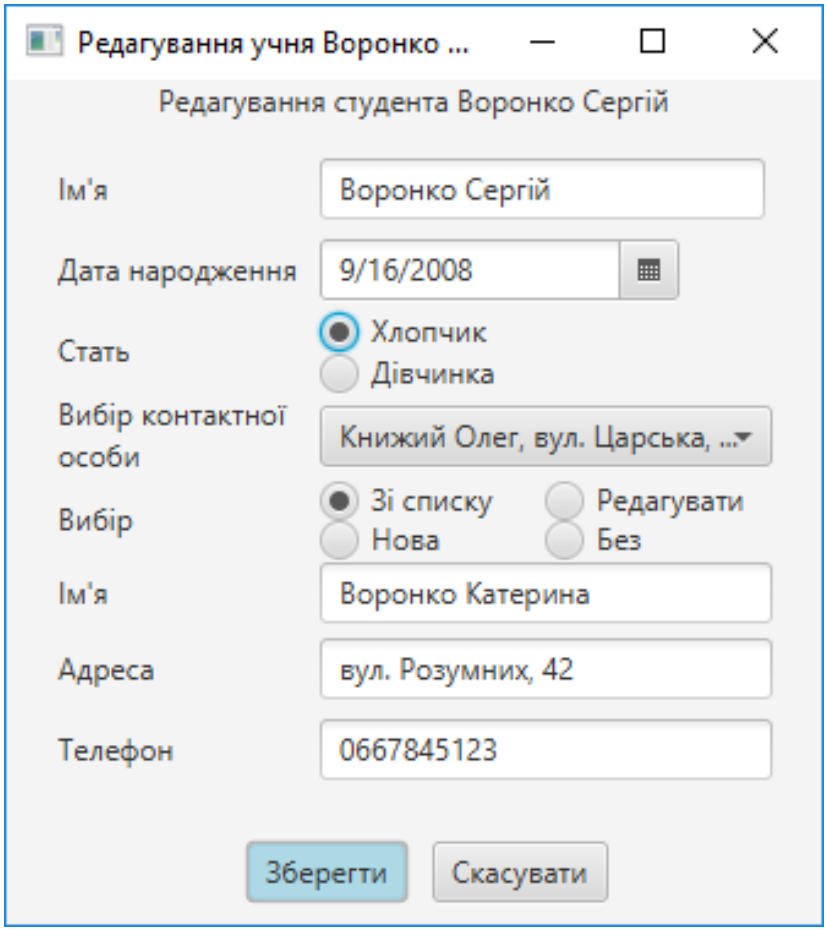


Рисунок 2.10 – Вікно додавання учня до групи.

Після редагування імені в текстовому полі, дня народження в вибирачі дати, вибору статі, вибору контактної особи та натисканні кнопки

«Зберегти», характеристики учня (ім’я, дата народження, стать, контактна особа) будуть замінені на записані.

Для відміни операції редагування учня в групі, натисніть кнопку «Скасувати» або на червону кнопку в верхньому кутку вікна.

2.3. Інструкція перегляду статистики та управління файлом БД

Вікно статистики учня

Для виклику вікна статистики учня є декілька способів:

- Кнопка «Інформація про учня» на вкладці «Журнал» головного вікна №9, рисунок 2.1.
- Кнопка «Інформація про учня» у вікні повного журналу групи №5, рисунок 2.3.
- Кнопка «Інформація про учня» у вікні редагування групи №2, рисунок 2.4.
- Кнопка «Інформація про учня» на вкладці «Управління групами» головного вікна №6, рисунок 2.5.

Вікно статистики учня не залежить від того, яким з цих способів воно було викликано (рисунок 2.10).

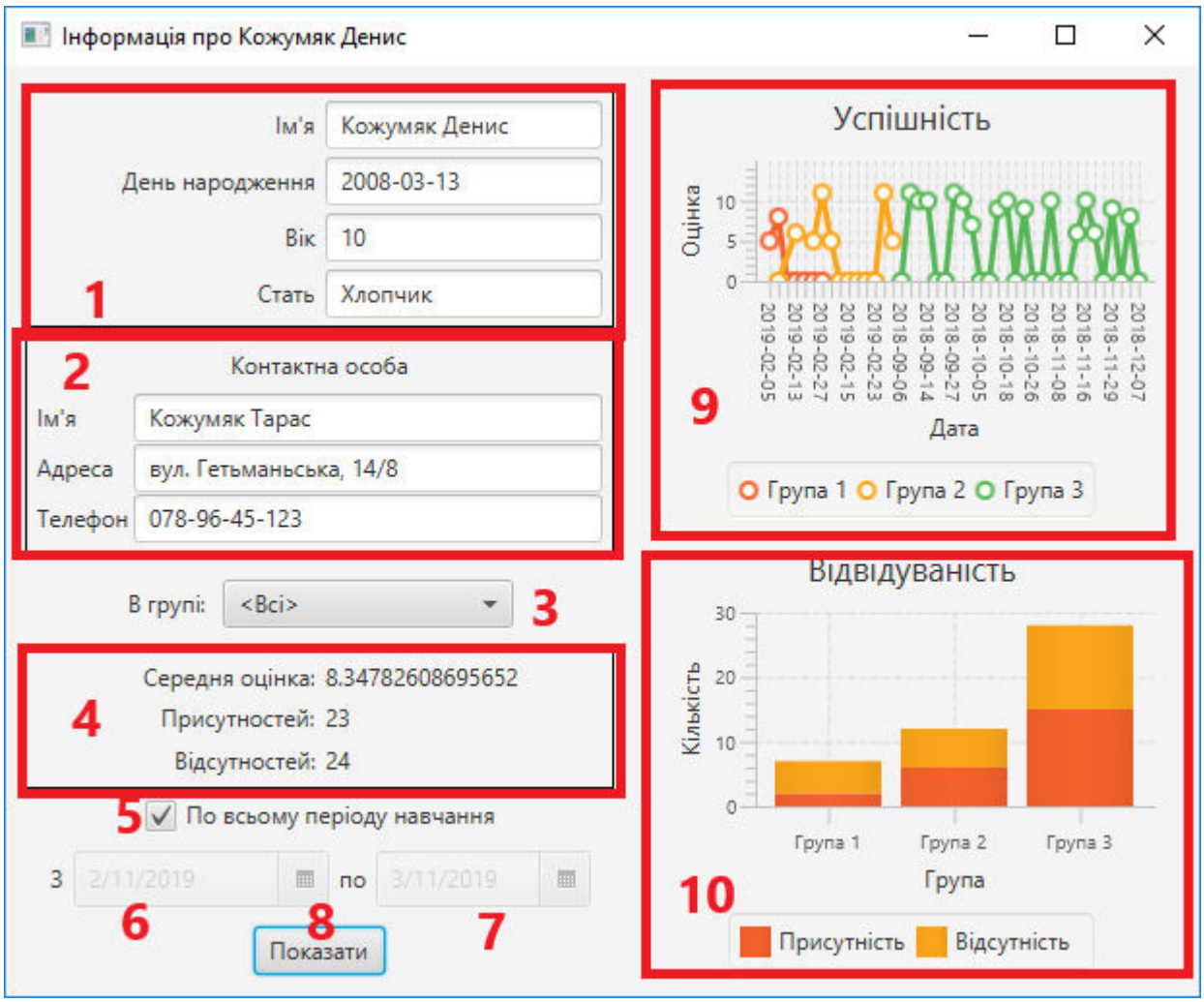


Рисунок 2.10 – Вікно інформації про учня

1 – Поле особистої інформації про учня.

2 – Поле контактної особи учня.

3 – Випадаючий список груп для перегляду успішності вибраного учня в даній групі.

4 – Поле інформації про успішність учня в вибраній в списку 2 групі.

5 – Прапорець вибору всього періоду навчання.

6 – Вибирач дати початку терміну, за який буде показуватися статистика.

7 – Вибирач дати кінця терміну, за який буде показуватися статистика.

8 – Кнопка оновлення поля статистики та графіків.

9 – Графік успішності.

10 – Графік відвідуваності.

У вікні статистики поле з особистою інформацією учня (№1) показує: ім’я, дата народження, вік та стать учня.

Випадаючий список №3 містить усі групи, в яких знайдено вибраного учня та запис «<Всі>». Програма шукає в усіх групах учня з таким ж ім'ям та датою народження. Так відбувається підбір груп. При виборі опції «<Всі>» буде показуватися статистика по всіх групах. При виборі певної групи, буде показуватися статистика по певній групі.

У полі статистики №4 зображується середній арифметичний бал (без врахування нулів, «null», якщо не знайдено записів оцінок, або всі - нулі), кількість присутностей та відсутностей у вибраній групі (у всіх групах, якщо вибрано в списку №3 варіант «<Всі>») за вибраний період (або за весь період, якщо відзначено прапорець №5).

Якщо прапорець №5 відмічено, то статистика буде показуватися по всьому часу.

Якщо прапорець №5 НЕ відмічено, то буде братися відрізок між датами, обраними в вибирачах дат №6 та №7 відповідно. Тобто, беруться до уваги тільки записи, зроблені пізніше дати в №6 і раніше дати в №7.

Кнопка «Показати», №8 оновлює поле статистики №4 та графіки №8 та №9.

Графік №9 – це лінійний графік оцінок учня. Кожна лінія – дані з певної групи (лінія одна, якщо обрано певну групу, декілька, якщо учень є в кількох групах і обрано в списку №3 варіант «<Всі>»). Кожна точка лінії – певний запис (в який день яку оцінку учень отримав). По осі абсцис (вісь X) знаходяться дати. По осі ординат (вісь Y) знаходяться оцінки.

Графік №10 – це графік присутностей. Якщо в списку №3 вибрано варіант «<Всі>», то графік – складаний стовпчиковий. Кожен стовпчик – дані по певній групі. Нижня частина стовпчика – присутності, верхня – відсутності. По осі абсцис (вісь X) знаходяться групи. По осі ординат (вісь Y) знаходяться кількість. Якщо ж в списку №3 вибрано певну групу, то графік

№10 – кругова діаграма з 2 компонентів: кількостей присутностей та відсутностей.

Вікна управління файлами бази даних.

Програма розрахована на використання різних файлів баз даних, якщо в них однакова внутрішня будова. Для управління базами даних, слід перейти на вкладку «Управління БД» головного вікна програми (рисунок 2.11).

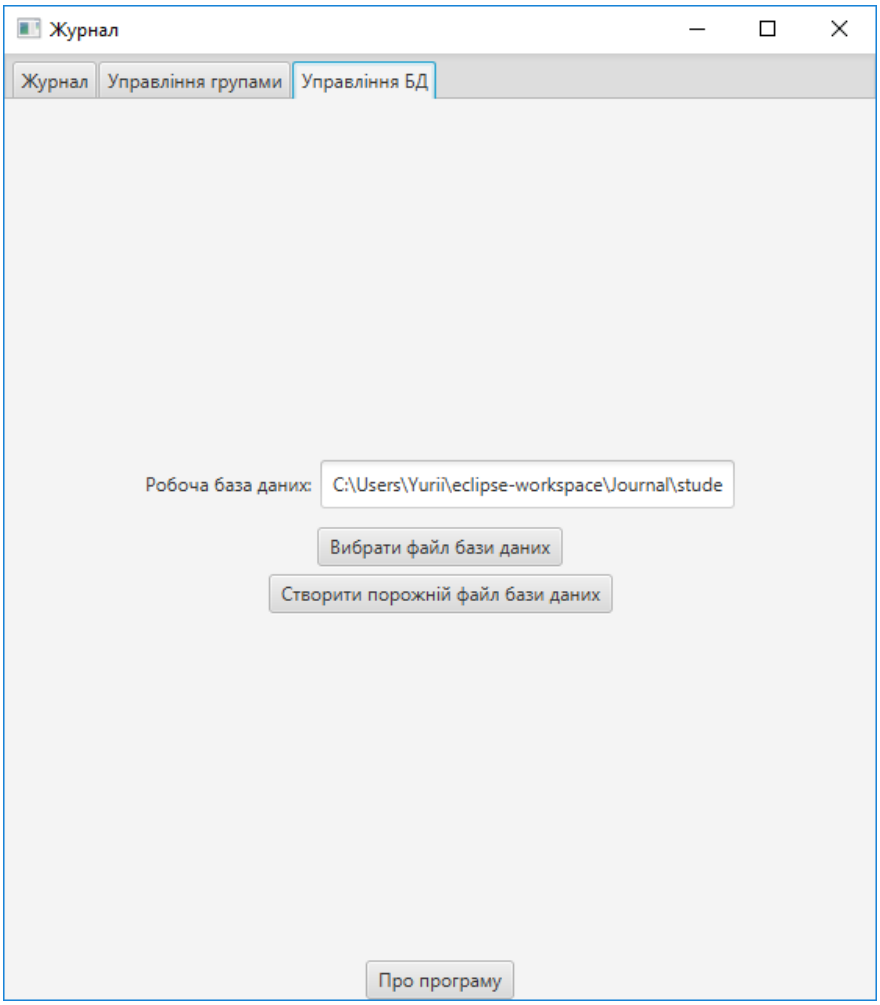


Рисунок 2.11 – Вкладка «Управління БД» головного вікна програми

В текстовому полі «Робоча база даних» зображено шлях до бази даних, яку в даний момент використовує програма.

При запуску програма спочатку шукає базу даних за замовчуванням і пробує до неї приєднатися.

База даних за замовчуванням – це файл students.db відповідної внутрішньої структури, який знаходиться в теці з фалом програми (*.jar-файлом програми).

Якщо програмі не вдалося найти базу даних за замовчуванням після запуску, вона покаже вікно попередження про незнаходження файлу БД за замовчуванням (рисунок 2.12) і запуститься без прив’язки до БД.

Зміна назви самої програми (*jar-файлу) на її роботу не впливає.

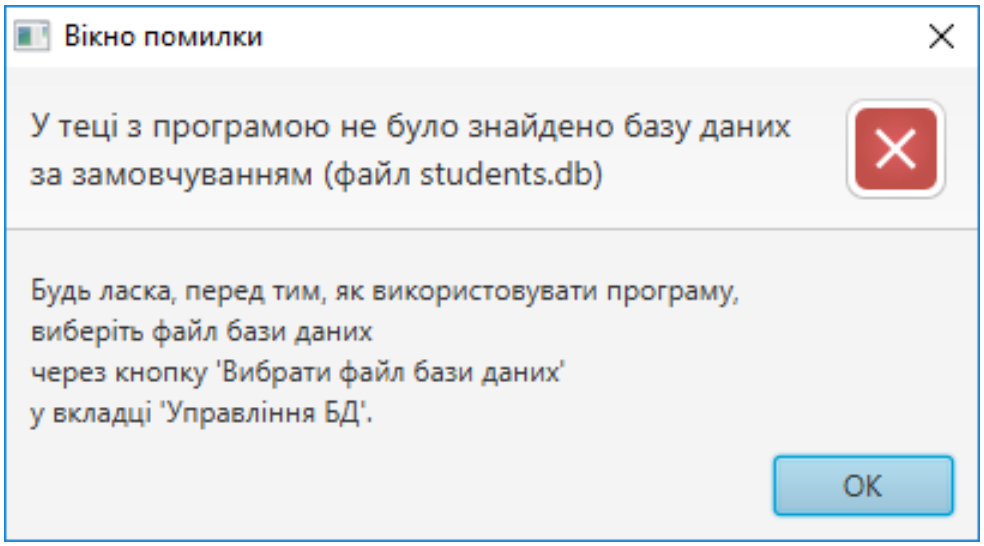


Рисунок 2.12 – Вікно попередження про незнаходження файлу БД за замовчуванням при старті програми

Для вибору нового файлу бази даних, необхідно натиснути кнопку «Вибрати файл бази даних», після цього обрати файл бази даних. Якщо було обрано коректний файл БД, тоді підключення нового файлу бази даних завершено.

Програма може працювати не з будь-яким файлом формату *.db, а лише з файлами формату *.db певної визначеної внутрішньої будови.

Для створення файлу бази даних необхідної структури без даних, слід натиснути кнопку «Створити порожній файл бази даних». Після цього, обрати теку і назву для нового файлу та зберегти його. Програма після створення нового порожнього файлу не змінює свій файл бази даних автоматично.

Використання окремих від програми фалів бази даних – можливість вести одну базу даних на кількох комп’ютерах та вести кілька баз даних на одному комп’ютері.

Для створення порожнього файлу бази даних, можна і скористуватися іншим способом:

– створити копію робочого файлу бази даних;

– підключити програму до копії файлу бази даних;

– видалити з бази даних усі групи.

Але, даний спосіб є небажаним для використання, адже окрім самих записів в базі даних, файл бази даних зберігає і іншу інформацію (порядок індексів і т.д.). Тому, створений таким способом файл бази даних матиме надлишкову інформацію, яка відсутня при створенні файлу бази даних через відповідну кнопку, використовуючи спеціальні функції програми.

Вікно інформації про програму

Натискання кнопки «Про програму» викликає вікно інформації про програму.

Вікно інформації про програму містить інформацію про розробника, замовника, версію, посилання на сторінку замовника та сторінку проекта програми на сайті Github і QR-код на дану сторінку.

Було зроблено рішення створити та утримувати сторінку проекту програми на сайті Github. Це дозволяє використовувати переваги системи Git, майже не витрачаючи час на розгортання системи Git. А саме такі переваги:

– система контролю версій (можливість скачати попередню версію програми. якщо зміни, внесені в остпнньому патчі приводять програму до неробочого стану)

– доступ до програми через мережу Інтернет (при потребі, кожен працівник організації може скачати програму з будь якої точки Землі)

– доступ до проекту через мережу Інтернет (при подальшій розробці чи внесення оновлень в програму, немає необхідностц при переході на новий комп'ютер переносити дані проекту на фізичних носіях. Достатньо лише зареєструватися на сайті Github, і вже все готово, щоб вносити зміни до проекту).

Таким чином, було розроблено користувацький інтерфейс програмного продукту для магістерської роботи. Даний інтерфейс покриває практично усі сценарії використання програмного продукту. Даний користувацький інтерфейс досить зручний в користуванні та інтуїтивно зрозумілий в багатьох випадках. Для навчання користування розробленим користувацьким інтерфейсом необхідно досить мала кількість часу, що являється ознакою того, що даний розроблений користувацький інтерфейс програмного продукту розроблено вдало.

При подальшій роботі над програмним продуктом, інтерфейс може змінитися. Можуть змінитися як і чисто візуальна частина інтерфейсу (наприклад, змінення кольору, переміщення елементів інтерфейсу в межах вікна, зміна форми та наповнення статичного тексту), так і функціональна частина інтерфейсу (додавання нових полів та забирання старих полів, додавання нових кнопок для реалізації нових функцій програми, не покритих розробленим інтерфейсом).

Зрештою, даний інтерфейс та середовище розробки інтерфейсу дозволяють досить легко проводити такі зміни.

3 ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

3.1 Загальний підхід до визначення економічної ефективності розробки

Обов’язковою складовою частиною будь-якого інжинірингового проекту, в тому числі софтверного, є фінансові витрати на різних етапах виконання робіт. Відповідно, важливо вірно здійснити фінансову оцінку передбачуваних витрат, продуктивність, корисність та, в результаті, економічну ефективність проекту.

Наукоємні розробки та дослідження, на відміну від корпоративних, не завжди супроводжують за мету отримання прибутку або ж іншої матеріальної вигоди. В багатьох випадках, проекти наукового спрямування не є економічно вигідними. Однак, вони є рушійною силою прогресу, дослідженням незвіданих галузей та проблем, які, у свою чергу, майже завжди впливають на майбутні різнопланові розробки. Тому дуже часто наукова діяльність стимулюється зовнішніми інвестиціями та підтримкою держаних інститутів, міжнародних грантів. В плані використання результатів досліджень та на основі отриманих моделей можна робити прогнози по впровадженню нових методик та принципів організації роботи діагностичних установ. Різноманітні медичні центри зможуть скористатися на практиці розробленою системою для покращення існуючих та отримання нових діагностичних методик.

Оцінка вартості дослідницьких розробок базується на витратному підході: використанні первісної вартості об'єктів, виходячи з фактичних витрат на розробку та доведення до комерційного використання з урахуванням амортизації. Так, як результати роботи у вигляді математичних моделей та реалізованого на їх основі ПЗ не буде використовуватися в комерційних цілях та не підлягатиме продажу, а становить наукову та інтелектуальну цінність, то доходу від продажі ПЗ та розробки як такого не передбачається. Іншими словами, всі вкладені кошти та витрати на розробку

даного рішення є не взаємоокупними, що несуть лише витрати у кількості залучених ресурсів та матеріальних засобів.

Згідно Статті 8 Закону № 3792-12 передбачено, що твори наукового характеру та комп'ютерні програми є об'єктами авторського права [25]. У разі реєстрації виконаної роботи як інтелектуальної власності та авторського права у державній службі інтелектуальної власності України необхідно буде сплатити збори за державну реєстрацію (382,5 грн.).

Для отримання відмінних результатів експериментів та доцільності розробки такого спеціалізованого ПЗ потрібні відповідні затрати на дослідження та розробку. Це і становитиме основу витрат, які будуть здійснені протягом підготовки та виконання реалізації даного рішення. Уявно модель витрат можна поділити на дві основні частини: витрати, пов'язанані на дослідження предметної області, побудову математичних моделей, отримання попередніх результатів експериментів, та частину реалізації програмної системи, архітектури та тестування.

Враховуючи залежність якості кінцевого продукту від кваліфікації програмістів, потрібно сконцентрувати увагу на якості та результативності розробки. Для цього потрібно провести ґрунтовний аналіз предметної області, залучити найновіші та інноваційні технології, провести ґрунтовне тестування та оцінку результатів розробки. Для забезпечення хорошої результативності при розробці доводиться йти на додаткові заходи заохочення та стимулювання у вигляді преміювання працівників, підтримання наукового дослідження досвідом іноземних науковців, дорогоцінних лабораторних дослідів.

До створення ПЗ можуть бути залучені позаштатні програмісти як зареєстровані, так і не зареєстровані підприємцями. В обох випадках співпраця з ними здійснюється на підставі цивільно-правового договору, найчастіше – договорами підряду. Щодо оподаткування виплат за договором підряду, то все залежить від того, чи зареєстрований виконавець підприємцем. Важливим етапом розробки ПЗ є його тестування, яке

виконується тестувальником за певну винагороду. Якщо тестувальники не перебувають з підприємством у трудових відносинах, оплата виконується на основі договору підряду на виконання робіт з тестування ПЗ. Сам же результат розробки не оподатковується, адже не є комерційним проектом і не спрямований на продаж.

3.2 Розрахунок вартості процесу розробки та оцінка економічної ефективності проекту

Для оцінки нематеріальних активів використовують міжнародні стандарти оцінки розрахунку вартості об’єктів інтелектуальної власності, розроблені TIAVIS (The International Assets Vabulation Standarts Commitie).

Виконання розробки програмного забезпечення з огляду економічної моделі можна виконувати двома способами: процедурним та об’єктно-орієнтованим. Обидва підходи потребують залучення ресурсів у вигляді програмісті-розробників, тестувальників, керівника проекту, наукового ресурсу. Різниця виникає в самій схемі розробки, тривалості періоду розробки та відповідній вартості. Процедурний підхід для розробки ПЗ в основі якого лежать процедури і функції передбачає розробку ПЗ як монолітного композиту, що в подальшому, як правило, вимагає великих витрат на супровід та модернізацію. Об’єктно-орієнтований підхід, що ґрунтується на основі об’єктів певних класів, що описують певну область, описують певну поведінку (методи) та володіють властивостями (атрибутами), орієнтовані на варіанти використання та покроковий процес розробки [26].

Для початку робіт необхідно скласти технічне завдання на розробку, яке є основним документом, що регламентує подальшу роботу, та містить докладний опис необхідних функцій програми, інтерфейс, технології, інше. Вартість складання технічного завдання переважно складає до 10% від

планованої вартості розробки. Роботу зі складання технічного завдання веде керівник проекту разом із програмістами та консультуючись із замовником.

Усі програмісти, що працюють у штаті підприємства-розробника мають встановлено певний посадовий оклад. Місячний оклад, денна заробітна плата, трудомісткість (днів) і основна заробітна плата кожного учасника техпроцесу представлено у таблиці 3.1. Всі суми наведені в національній валюті – в гривні.

Таблиця 3.1 – Розрахункова вартість технологічного процесу розробки

Посада	Місячний оклад, грн.	Денна зар. плата, грн.	Об'єктно-орієнтований підхід		Процедурний підхід	
			Днів	Сума, грн.	Днів	Сума, грн.
Тімлід	9900,00	450,00	20	9000,00	25	11250,00
Програміст	9570,00	435,00	32	13920,00	35	15225,00
Розробник тестів	8470,00	385,00	5	1925,00	6	2310,00
Дизайнер	8800,00	400,00	7	2800,00	7	2800,00
Додаткова зар. плата 20%			44	5529,00	48	6317.00
Фонд оплати праці 36,77%				10165,07		11613,80
Всього витрат на зар. плату				43339,07		49515,80
Військовий збір 1,5%				650,09		742,74
Єдиний соціальний внесок 3,6%				1560,21		1782,57
ПДВ, 15%				6500,86		7427,37
Всього				52050,22		59468,48

Згідно вимог та прорахованої кількості необхідних ресурсів на виконання, розробку, тестування та дослідницьку роботу було отримано

основні часові рамки роботи над проектом. Так для об’єктно-орієнтованого підходу загальна тривалість роботи над ПЗ становить 44 робочих днів (під робочим днем розуміється 8-ми годинний робочий день), що включає роботу програміста, який, в свою чергу, являється і керівником розробки, роботу тестувальника та наукового працівника. Сума витрат на заробітню плату становить 52050,22 гривень включаючи всі види додаткових оплат. Для процедурного підходу до розробки суми дещо більші, адже затрачається більше часу на розробку. Так, при використанні процедурного підходу сумарна тривалість часу розробки становить 48 робочі дні, та витрати у вигляді виплат заробітної плати становлять 59468,48 гривень.

Витрати на науково-дослідницьку роботу та здійснення розробки програмних продуктів і об’єктно-орієнтованим, і процедурним способом включають:

Основна заробітна плата:

$ЗП_{осн\ 1}$ = 27645 грн; $ЗП_{осн\ 2}$ = 31585 грн.

Додаткова заробітна плата обчислюється як $ЗП_{дод} = 0,2 \cdot ЗП_{осн}$.

$ЗП_{дод\ 1}$ = 0,2 × 27645 грн= 5529 грн; $ЗП_{дод\ 2}$ = 0,2 × 31585 грн = 6317 грн.

Нарахування на фонд оплати праці (ФОП):

$ФОП_{ЄСВ}$ = 0,3677·× ФЗП

$ФОП_{ЄСВ1}$ = 0,3677·× 33174 грн = 10165,07 грн;

$ФОП_{ЄСВ1}$ = 0,3677·× 37902 грн = 11613,80 грн.

Всього витрат:

$В_{ЗП1} = ЗП_1 + ФОП_{ЄСВ1} + ЗП_{дод1}$ = 43339,07 грн;

$В_{ЗП2} = ЗП_2 + ФОП_{ЄСВ2} + ЗП_{дод2}$ = 49515,80 грн.

З цієї суми утримуються обов’язкові відрахування на заробітну плату: єдиний соціальний внесок, який складає 3,6% від суми нарахованої заробітної плати та податок на доходи фізичних осіб, який складає 15% від суми нарахованої заробітної плати, зменшеної на суму єдиного внеску на загальнообов’язкове соціальне страхування та податкової соціальної пільги, військовий збір у розмірі 1,5%, від суми нарахувань.

До окремих витрат також відносяться витрати на куповані вироби (матеріальне забезпечення) та спец обладнання для підтримки експерименту, накладні витрати. Витрати, що будуть супроводжувати проект розробки, порівнюватимемо в двох можливих підходах розробки.

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни (формула 3.1).

$$M_{Bi} = q_i \cdot p_i \ , \tag{3.1}$$

де q_i – кількість витраченого матеріалу i-го виду; p_i – ціна матеріалу i-го виду.

Матеріальні витрати в рамках проекту наведені в таблиці 3.2. Загальна сума матеріальних витрат становить 2188 гривень.

Таблиця 3.2 – Матеріальні витрати

Найменування ресурсу	Кількість, шт.	Ціна одиниці, грн	Загальна сума, грн
Флешки	5	250,00	1250,00
Папір для друку А4, арк	1000	0,178	178,00
Тонер для принтера	2	80,00	160,00
Дошка для записів	1	500,00	500,00
Перманентний маркер	10	10,00	100,00
Всього			2188.00

Розрахунок витрат на електроенергію одиниці обладнання визначаються за формулою 3.2:

$$З_в = W * T * S, \tag{3.2}$$

де W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії, S = 2,50 грн/кВт·год.

$З_{в1}$ =1,2 * 352 * 2,50 = 1056,00 грн;

$З_{в2}$ =1,2 * 384 * 2,50 = 1152,00 грн;

Розрахунок суми амортизаційних відрахувань. Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 %, вартість яких перевищує 1000 грн. і визначається:

$$A = \frac{C_Б \cdot N_A \cdot T_{ФАК}}{T_{ГОД}}, \quad (3.3)$$

де $C_Б$ – балансова вартість обладнання, грн; N_A – норма амортизаційних відрахувань в рік, %; $T_{ФАК}$ – фактичний час роботи обладнання по написанню програми, год; $T_{ГОД}$ – річний робочий фонд часу, год.

У даній формулі норма відрахувань на амортизацію рівна N_A = 0,6. Балансова вартість обладнання вказана в таблиці 3.3 і рівна $C_Б$ = 37800 гривень. Річний робочий фонд часу приймемо за $T_{ГОД}$ = 2112 годин. З них реальний фактичний робочий час становить $T_{ФАК}$ = 352 години згідно об'єктно-орієнтованого підходу та $T_{ФАК}$ = 384 годин згідно процедурного підходу.

Згідно вищезгаданої формули витрати на амортизацію становлять 3765,74 гривень та 4108,08 гривень для кожного підходу відповідно.

Таблиця 3.3 – Перелік необхідного обладнання

Найменування	Кількість, шт	Ціна, грн	Сума, грн
Комп'ютер	3	11200,00	33600,00

Принтер	1	4200,00	4200,00	
Середовища розробки	3	безкоштовно	безкоштовно	
Операційна система (Linux)	3	безкоштовно	безкоштовно	
Всього більше 1000 грн.			37800,00	
Всього витрат на амортизацію			3765,74	4108,08
Всього			41565,74	41908,08

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління та створення необхідних умов праці та закупівлю ресурсів та обладнання для розробки наведені в таблиці 3.3.

Залежно від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників. Нехай вона буде дорівнювати 40%, що становить 20820,09 грн для об’єктно-орієнтованого і 23787,39 грн для процедурного підходу розробки.

Проведемо розрахунок вартості створюваного програмного продукту. Вартість продукції включає у собі собівартість і планований прибуток. Найважливішим моментом для розробника, з економічної точки зору, є процес встановлення ціни.

Можна отримати загальні значення витрат на розробку та реалізацію проекту, враховуючи всі вище описані затрати та нарахування. Цей вид витрат складається з сум витрат на оплату праці (всього витрати на оплату праці), матеріальні затрати, затрати на електроенергію, накладні витрати, витрати на обладнання, враховуючи амортизації обладнання на час виконання проекту.

Собівартість продукції – це сума грошових витрат підприємства (фірми) на виробництво і збут одиниці продукції, виконання робіт та надання послуг.[27]

Повна собівартість програмного продукту дорівнює сумі усіх витрат на його виробництво: 117671,59 грн використовуючи об’єктно-орієнтований підхід, 128494,73 грн при процедурному підході розробки.

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність (Е) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E = \frac{П}{Св}, \tag{3.4}$$

де *П* – прибуток, *П= В - Св; Св* – собівартість.

У випадку даної розробки, маючи некомерційний проект без економічно корисного результату, можна прогнозувати, що економічна ефективність прямує до 0 у обох випадках. Однак це не є причиною для негативного економічного висновку щодо даного проекту, адже такого плану розробки приносять користь у вигляді інтелектуальних ресурсів, і, переважно, фінансуються або виконуються на замовлення організацій, зацікавлених в отриманні результатів досліджень та розробок. Фінансування не можна вважати отриманим доходом від реалізації. Однак за надходження коштів на реалізацію ззовні можна вважати ефективність проекту рівною 1 (Е = 1), що означає перекриття витрат на розробку у повній мірі, тобто фінансування.

Якщо ринкова вартість програмного продукту рівна прийнятій, то економічна ефективність визначається встановленим рівнем прибутку. Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($T_{ок}$):

$$T_{ок} = \frac{1}{E} \tag{3.5}$$

У нашому випадку прямого прибутку не існує. Прибуток можна прогнозувати на підприємствах, організаціях чи відомствах, що зацікавлені в дослідженні. Окупність же для розробки даного ПЗ за ефективності рівній одиниці можна вважати теж рівній 1 згідно формули 3.4.

Виходячи із експертних оцінок і складності програми, приймемо величину витрат на супровід і модернізацію програмного забезпечення, створеного за об'єктно-орієнтованим методом 25% (29420,01 грн) від початкових витрат, а за процедурним – 30% (38551,18 грн).

Однак варто зауважити, що розробка спрямована на короткотривалу підтримку та не прогнозує модернізації. У разі ж необхідності розробки такого ж або суміжного ПЗ можна вважати за доцільно розпочинати розробку з початкових етапів, що потягне за собою нові витрати у повній мірі. Тобто у разі короткотермінової підтримки все ж за доцільніше обирати об'єктно-орієнтовану модель розробки, адже вартість короткотермінової підтримки згідно цієї моделі не вплине значно на сукупну вартість розробки.

Також не зайвим буде вказати на те, що пир використанні об'єктно орієнтованої моделі розробки дозволить в подальшому легше додавати новий функціонал до програмного продукту. Також розроблені модулі програми можуть бути повторно використані в подальших проектах, що зменшить витрати на розробку подальших програмних продуктів.

Здана в експлуатацію система не завжди цілком завершена, її треба змінювати протягом терміну експлуатації. Внаслідок змін система стає більш складною і погано керованою. Об'єктно-орієнтоване представлення програми дозволяє навіть середньому програмісту швидко і ефективно супроводжувати і модернізувати програми, що значно скорочує подальші витрати на супровід і модернізацію [27].

Сумарні дані економічного розрахунку розробки даного проекту наведені в таблиці 3.4.

Таблиця 3.4 – Загальні витрати

Вид витрат	Об'єктно-орієнтований підхід, грн	Процедурний підхід, грн
Зарплата основна	27645,00	31585,00
Зарплата додаткова	5529,00	6317,00
Фонд заробітної плати	33174,00	37902,00
Відрахування на ФОП	10165,07	11613,80
Разом на оплату праці	43339,07	49515,80
Матеріальні витрати	2188,00	2188,00
Електроенергія	1056,00	1152,00
Амортизація	3765,74	4108,08
Накладні витрати	20820,09	23787,39
Разом на інші витрати	27829,82	31235,47
Обладнання	41052,23	41373,17
Собівартість	117680,04	128503,95
Прибуток	відсутній	відсутній
Вартість розробленого ПЗ	**117680,04**	**128503,95**
Модернізація і супровід	29420,01	38551,18
Загальні витрати на розробку	147100,05	167055,13
Економія ($ЗВ_1$-$ЗВ_2$)		**19955,08**

Загальна вартість пропонованих робіт становить 147100,05 гривень для процедурного і 167055,13 гривень для об'єктно-орієнтованого підходів розробки. В даному випадку реалізації проекту варто вибрати об'єктно орієнтований підхід для розробки даного ПЗ, адже фінансово це більш вигідно. Також у даній методиці розробки кращі часові рамки та перспективи підтримки і модернізації. У оцінці вартості продукту варто враховувати можливість фінансування та спонсорування проекту, що дозволить гнучко та ефективно підійти до розробки та організації праці.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Розглянемо стандартне робоче комп'ютеризоване місце для працівника, який проводить за комп'ютером більшу частину робочого часу за розробкою інформаційно-електронної системи для контролю відвідуваності та успішності студентів «iJournal» на основі технології JavaFX. Точніше, робоче місце середньої важкості IIб за ДСН 3.3.6.042-99 з ВДТ [28].

Роботодавець поінформував працівників під розписку про умови праці та наявність на наших робочих місцях небезпечних та шкідливих виробничих факторів (фізичних, хімічних, біологічних, психофізіологічних), які виникають під час роботи з екранними пристроями та ще не усунуто, а також про можливі наслідки їх впливу на здоров'я працівників відповідно до вимог статті 5 Закону України „Про охорону праці", тобто робоче місце відповідає ДСанПІН 3.3.2.007-98 [29].

Роботодавець забезпечив навчання і перевірку знань працівників з питань охорони праці та безпечного використання екранних пристроїв до початку роботи з ними, а також у випадках модифікації та організації роботи обладнання, тобто робоче місце відповідає ДСанПІН 3.3.2.007-98.

Робочого місця працівника з екранними пристроями обиране з таким устаткування, яке не створює зайвого шуму та не виділяє надлишкового тепла. Рівні шуму на робочих місцях осіб, які працюють з екранними пристроями, відповідають вимогам Санітарних норм виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99.

Роботодавець за рахунок тривалості робочої зміни організує внутрішні регламентовані перерви для відпочинку відповідно до Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98.

Роботодавець забезпечує за свій рахунок проведення медичних оглядів працівників відповідно до вимог Порядку проведення медичних

оглядів працівників певних категорій, затвердженого наказом Міністерства охорони здоров'я України від 21 травня 2007 року № 246, зареєстрованого в Міністерстві юстиції України 23 липня 2007 року за № 846/14113 [30].

В приміщенні з робочими місцями середня температура повітря 20 °С, вологість повітря 50%, швидкість повітря 0.2 м/с, тобто воно відповідає п.1.1.1 ДСН 3.3.6.042-99.

Температура внутрішніх поверхонь робочої зони (стіни, підлога, стеля), технологічного обладнання (екрани і т. ін.), зовнішніх поверхонь технологічного устаткування, огороджуючих конструкцій не виходить більш ніж на 2 °С за межі оптимальних величин температури повітря для даної категорії робіт, тобто приміщення відповідає п.1.1.2 ДСН 3.3.6.042-99.

В холодний період року в приміщеннях з робочими місцями температура повітря +20 °С, відносна вологість 75%, швидкість руху повітря – 0.2 м/с. В теплий період року в приміщеннях з робочими місцями температура повітря +25 °С, відносна вологість 70%, швидкість руху повітря – 0.3 м/с. Тобто, робоче місце відповідає п.1.2.2 ДСН 3.3.6.042-99.

Температура внутрішніх поверхонь приміщень (стіни, підлога, стеля), а також температура зовнішніх поверхонь технологічного устаткування або його захисних обладнань (екранів і т. ін.) не виходить за межі допустимих величин температури повітря для даної категорії робіт, тобто відповідає п.1.2.5 ДСН 3.3.6.042-99.

Так як приміщення з робочими місцями є зі значними площами засклених поверхонь, в ньому передбачені заходи щодо захисту від перегрівання при попаданні прямих сонячних променів в теплий період року (орієнтація віконних прорізів схід - захід, улаштування жалюзі та ін.), від радіаційного охолодження - в зимовий (екранування робочих місць). Тобто, робоче місце відповідає п.2.3 ДСН 3.3.6.042-99.

Проводяться попередні (при прийомі на роботу) та періодичні медичні огляди в процесі роботи відповідно з діючим наказом МОЗ України. Тобто, робоче місце відповідає п.2.13 ДСН 3.3.6.042-99.

Вимірювання параметрів мікроклімату проводяться на робочих місцях і в робочій зоні на початку, в середині та в кінці робочої зміни. Тобто, робоче місце відповідає п.3.1 ДСН 3.3.6.042-99.

Вимірювання здійснюються не менше 2-х разів на рік (теплий та холодний періоди року) у порядку поточного санітарного нагляду, а також при прийманні до експлуатації нового технологічного устаткування, внесенні технічних змін в конструкцію діючого устаткування, організації нових робочих місць тощо. При проведенні вимірювання в холодний період року температура зовнішнього повітря не вища за середню розрахункову температуру, в теплий період - не нижча за середню розрахункову температуру, що приймається для опалення та кондиціонування за оптимальними та допустимими параметрами. Тобто, робоче місце відповідає п.3.2 ДСН 3.3.6.042-99.

Вимірювання параметрів мікроклімату на робочих місцях по розробці інформаційно-електронної системи для контролю відвідуваності та успішності студентів «iJournal» на основі технології JavaFX проводяться на висоті 0,5 - 1,0 м від підлоги - при роботі сидячи, 1,5 м від підлоги - при роботі стоячи. Тобто, робоче місце відповідає п.3.3 ДСН 3.3.6.042-99.

Температура та відносна вологість повітря вимірюються приладами, заснованими на психрометричних принципах. Можливе використання тижневих і добових термографів і гігрографів. Тобто, робоче місце відповідає п.3.6 ДСН 3.3.6.042-99.

Швидкість руху повітря вимірюється анемометрами ротаційної дії. Малі величини швидкості руху повітря (менше 0,3 м/сек.), особливо при наявності різноспрямованих потоків, вимірюються електроанемометрами, циліндричними або кульовими кататермометрами. Тобто, робоче місце відповідає п.3.7 ДСН 3.3.6.042-99.

Температура поверхонь огороджуючих конструкцій (стін, стелі, підлоги) або обладнань (екранів і т. ін.), зовнішніх поверхонь технологічного устаткування вимірюються приладами, що діють за принципом

термоелектричного ефекту. Тобто, робоче місце відповідає п.3.8 ДСН 3.3.6.042-99.

Приміщення для роботи з ЕОМ розташовані не у підвальних приміщеннях та не на цокольних поверхах. Тобто, робоче місце відповідає п.2.2 ДСанПіН 3.3.2-007-98.

Площа на одне робоче місце становить 8 $м^2$, а об'єм – 25 $м^3$. Тобто, робоче місце відповідає п.2.3 ДСанПіН 3.3.2-007-98.

Природне освітлення здійснюється через світлові прорізи, орієнтовані переважно на північ та північний схід і зебезпечують коефіцієнт природною освітленості (КПО) 3%. Тобто, робоче місце відповідає п.2.5 ДСанПіН 3.3.2-007-98.

Віконні прорізи приміщень для роботи з ВДТ обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки). Тобто, робоче місце відповідає п.2.9 ДСанПіН 3.3.2-007-98.

Робоче приміщення оснащені аптечками першої медичної допомоги. Тобто, робоче місце відповідає п.2.16 ДСанПіН 3.3.2-007-98.

Зазначення освітлення освітленості на поверхні робочого столу в зоні розміщення документів становить 400 лк. Тобто, робоче місце відповідає п.3.2.3 ДСанПіН 3.3.2-007-98.

Конструкція робочого місця забезпечує підтримання оптимальної робочої пози. Тобто, робоче місце відповідає п.4.2 ДСанПіН 3.3.2-007-98.

В заключення можна сказати, що розглянуті стандарти охорони праці на робочих місцях розробки інформаційно-електронної системи для контролю відвідуваності та успішності студентів «iJournal» на основі технології JavaFX відповідає усім розглянутим основним нормативам для робочих місць даного типу.

4.2. Забезпечення безпеки життєдіяльності при роботі з персональним комп’ютером

Комп’ютери вже давно стали важливим елементом оточення більшості людей. Комп’ютери використовуються і для офісної роботи, і для дистанційної роботи з дому, і для розваг, і для спілкування та багато іншого. А так як робота за комп’ютером може мати багато негативних факторів на користувача ПК, дослідження безпеки життєдіяльності при роботі з персональним комп’ютером є дуже важливим. Особливо це стосується розробників, адже практично вся їхня робота пов’язана тим чи іншим чином з використанням ПК.

Варто ще зазначити, що вдосконалення цифрових технологій часто вносять зміни в техніку безпеки використання ПК. Наприклад, впровадження LED та LCD дисплеїв комп’ютерів на заміну дисплеїв з електронно-променевою трубкою, дозволило зменшити негативний вплив дисплею ПК на очі користувача, тобто збільшити безпечно можливий час роботи за ПК. Натомість, впровадження лазерних принтерів на заміну струйним додало вимогу до вентиляції озону з приміщення, так як лазери можуть генерувати цей шкідливий газ.

Заходи щодо усунення небезпеки ураження електричним струмом зводяться до правильного розміщення устаткування та електричних кабелів. Інші заходи щодо забезпечення електробезпеки, збігаються з загальними заходами пожежо- та електробезпеки.

В якості профілактичних заходів для забезпечення пожежної безпеки слід використовувати приховану електромережу, надійні розетки з пожежобезпечних матеріалів, силові мережі живлення устаткування виконувати кабелями, розрахованими на підключення в 3-5 разів більшого навантаження, включати й виключати живлення обладнання за допомогою штатних вимикачів. Треба регулярно робити очистку внутрішніх частин комп'ютерів, іншого устаткування від пилу, розташовувати комп'ютери на

окремих неспалюваних столах. Для запобігання іскріння необхідно рідше встромляти і виймати штепсельні вилки з розеток.

Екран дисплея повинен бути розташованим перпендикулярно до напрямку погляду. Якщо він розташований під кутом, то стає причиною сутулості. Відстань від дисплея до очей повинна трохи перевищувати звичну відстань між книгою та очима. Перед екраном монітора, особливо старих типів, повинен бути спеціальний захисний екран. При його відсутності треба сидіти на відстані витягнутої руки від монітора. Ще одним моментом, який стосується зору, є необхідність створення неоднорідного поля зору. Для цього можна розвісити на поверхнях (стінах) плакати та картини, виконані у спокійних тонах. Наприклад, пейзажі.

Важливою є форма спинки крісла, яка повинна повторювати форму спини. Висота крісла повинна бути такою, щоб користувач не почував тиску на куприк або стегна. Крісло бажано обладнати бильцями. Його потрібно встановити так, щоб не треба було тягтися до клавіатури. Періодично користувачу необхідно рухатися, вчасно змінювати положення тіла і робити перерви у роботі.

При напруженій роботі за комп'ютером щогодини необхідно робити перерву на 15 хвилин через кожну годину і треба займатися іншою справою. Декілька разів на годину бажано виконувати серію легких вправ для розслаблення.

Наслідками регулярної роботи з комп'ютером без застосування захисних засобів можуть бути: захворювання органів зору (60% користувачів); хвороби серцево-судинної системи (20%); захворювання шлунково-кишкового тракту (10%); шкірні захворювання (5%); різноманітні пухлини.

Режим праці та відпочинку при роботі з персональною електронно-обчислювальною машиною (ПЕОМ) залежить від категорії трудової діяльності. Всі роботи з ПЕОМ ділять на три категорії. Перша – епізодичне зчитування і робота з інформацією не більше 2-х годин за 8-годинну робочу

зміну. Друга – зчитування інформації або творча робота не більше 4-х годин за восьми годинну зміну. Третя – зчитування інформації або творча робота тривалістю більше 4-х годин за зміну.

Якщо у приміщенні експлуатується більше одного комп'ютера, то треба врахувати, що на користувача одного комп'ютера можуть впливати випромінювання від інших, в першу чергу бокових, а також і задньої стінки сусіднього дисплея. Тому необхідний захист спеціальними фільтрами і щоб користувач розміщався від бічних і задніх стінок інших дисплеїв на відстані не ближче одного метра.

Отже, щоб запобігти негативним впливам необхідно знати й небезпечні сторони самого комп'ютера і правила безпечної роботи, знати засоби запобігання небезпек. Вони пов'язані перед усім із загально відомими небезпечними факторами – поразками електричним струмом, пожежонебезпечністю.

Негативні фактори впливу на здоров'я. Всесвітня організація охорони здоров'я (ВООЗ) роботу з персональним комп'ютером віднесла до небезпечних, бо їй притаманний фактор постійно діючого стресу. Через це небезпеці піддаються всі життєво важливі органи людини, з'являється ризик виникнення серйозних хвороб.

Електромагнітні поля біля комп'ютера (особливо низькочастотні) негативно впливають на людину і в першу чергу на її центральну нервову систему, викликаючи головний біль, запаморочення, нудоту, депресію, безсоння, відсутність апетиту, виникнення синдрому стресу. Причому нервова система реагує навіть на короткі за тривалістю впливи слабких полів: змінюється гормональний стан організму, порушуються біоструми мозку. Це призводить до погіршення зору, ускладненню серцево-судинних захворювань, зниженню імунітету, виникають негативні впливи на плин вагітності.

Характерною рисою професії оператора ПК є статичний режим роботи: великий обсяг праці треба виконувати в сидячому положенні. При

цьому більшість груп м'язів постійно напружені, що призводить до швидкої стомлюваності, сприяє розвитку фахових патологічних вигинів хребта: грудному гіперкифозу, сплощенню шийного лордозу і формуванню сколіозів. Неправильне розташування дисплеїв по висоті – занадто низьке або високе, під неправильним кутом – є головною причиною появи сутулості. Занадто високе розташування дисплея призводить до тривалої напруги шийного відділу хребта, що, зрештою, може призвести до розвитку остеохондрозу. Ненормальний стан хребта може стати причиною захворювання всього організму.

Тепер розглянемо основні "комп'ютерні" хвороби. Нерухома напружена поза оператора призводить до втоми і виникнення болю в хребті, шиї, плечових суглобах. Інтенсивна робота з клавіатурою викликає болючі відчуття в ліктьових суглобах, передпліччях, зап'ястях і пальцях рук.

У деяких операторів персонального комп'ютера (ПК) розвивається м'язова слабкість, відбувається зміна форми хребта (синдром тривалого статичного навантаження – СТСН), що може призвести до непрацездатності. Постійні користувачі ПК найчастіше піддаються психічним стресам, хворобам серцево-судинної системи і верхніх дихальних шляхів. Значному навантаженню піддається зоровий апарат.

При тривалій та інтенсивній роботі за комп'ютером з'являється синдром комп'ютерного стресу (СКС), який проявляється головним болем, запаленням очей, алергією, дратівливістю, млявістю і депресією, погіршенням зосередженості і працездатності.

Причинами різноманітних симптомів СКС є 5 основних чинників: неправильна робота очей і поза тіла, носіння невідповідних окулярів або контактних лінз, неправильна організація робочого місця, розподілення фізичних, розумових, візуальних навантажень, низький рівень візуальної підготовленості для роботи з комп'ютером. Особливо це характерно для дітей, молодших школярів.

Існує і медико-соціальна проблема – комп'ютер і здоров'я дітей. Тепер в світі існує потужна індустрія виробництва комп'ютерних ігор. Діти з великим задоволенням віддають цим гарним і захоплюючим, а часто і агресивним іграшкам свій вільний час. Діти в значно меншому ступені, чим дорослі, спроможні контролювати себе. їхня психіка дуже нестійка, тому надмірне захоплення комп'ютерними іграми може стати причиною дуже важких наслідків – розвивається підвищена збудженість, у школярів знижується успішність, діти стають примхливими, некерованими, перестають будь-чим цікавитися крім комп'ютера.

Тунельний синдром зап’ястного каналу – запалення медіального нерва руки через набряк сухожиль, синовиальної оболонки.

Для зменшення шкідливого впливу клавіатури фірма Microsoft розробила ергономічну клавіатуру, що своєрідною формою, конструкцією знижує навантаження на руки. Творці клавіатури сподіваються, що вона застрахує оператора від тунельного синдрому зап'ястного каналу.

Інший пристрій, який привертає до себе увагу фахівців в області ергономіки – маніпулятор типу "миша". На жаль, навіть найерго-номічніша клавіатура не може цілком вирішити проблему, оскільки причини захворювання "травми повторюваних навантажень" дотепер цілком не виявлені.

Всесвітня мережа Інтернет вже охопила великі обсяги виробництва, банківську справу, наукові дослідження, освіти – від середньої до вищої. За допомогою глобального інтернету спілкується населення Землі. Виникає єдиний організм з спільною економікою, культурою, політикою, наукою, спільним інформаційним полем. Надалі ці інтегруючі процеси будуть наростати, але це може мати і негативні наслідки, зокрема з огляду загальної, індивідуальної безпеки.

Отже, щоб цьому запобігти потрібна висока компютерно-інформаційна грамотність, і перед усім спеціалістів у будь-якій сфері діяльності – промисловій, науковій, навчальній.

ВИСНОВКИ

Результати дослідження у магістерській роботі дають можливість зробити узагальнюючі висновки, що мають теоретичне та практичне значення, а саме:

1. Розроблені основні вимоги до структури і функціонування електронної системи для контролю відвідуваності та успішності студентів забезпечили використання комп'ютера в якості потужного засобу навчання (автоматизовані системи навчання, контроля знань і управління учбовим процесом) та адаптації інформаційних технологій навчання в умови навчального процесу.
2. Спроєктована програмна система дозволить покращити оснащення освітніх закладів засобами інформаційних технологій, посилити їх використання в якості нового педагогічного інструменту та забезпечити підтримку процесу навчання з метою його подальшого ефективного використання в професійній діяльності: раціональне використання часу вчителів/викладачів забезпечить можливість більше часу витрачати на навчання та підготовку учнів/студентів тому, що зекономиться час котрий тратився на заповнення паперового журналу.
3. Зконструйоване програмне забезпечення активує розвиток інформатизації суспільства, а також формує об'єднання переваг традиційної освіти з можливостями інформаційних технологій, які сприяють активізації пізнавальної діяльності учнів/студентів та підвищенню їх мотивації до навчання. Ширший функціонал електронного журналу, в порівнянні з паперовим, забезпечить якісніший контроль успішності для самих учнів/студентів та посилить здорову конкуренцію серед них.
4. Розроблена інструкція використання програмного продукту дозволяє створення єдиного інформаційного освітнього простору,

який забезпечить доступність якісної інформації, що, в свою чергу, сприятиме мінімізації корупційної складової в освітянській сфері, адже оцінки будуть виставлятися он-лайн, та усуне можливість помилки при ручному обчисленні середньої оцінки за модуль та кількості відвідувань завдяки автоматичному підрахунку даних.

Перехід навчальних закладів з паперового журналу на електронний дозволить використовувати велику кількість додаткових функцій журналу. Наприклад, електронний журнал обліку успішності та відвідуваності студентів може мати функцію перевірку поточної загальної успішності та відвідуваності студента за певний період, а не лише в кінці модуля. А це допоможе студенту аналізувати свою продуктивність по окремій дисципліні. Також ця функція може допомогти батькам учнів проводити нагляд за успішністю та відвідуваністю своїх дітей-учнів.

Можливість генерувати графіки, що відображають успішність та відвідуваність учнів/студентів дозволить наочно представляти ефективність кожного.

Зберігання даних журналу в електронному вигляді, а не в паперовому дозволить зберігати дані в мережі Інтернет, що дасть можливість вчителю/викладачу проводити роботу з журналом без прив’язки до місця роботи та паперової версії журналу. Також це дозволить учням/студентам та їх батькам переглядати навчальну успішність та відвідуваність уроків дистанційно, навіть зі свого дому.

Таким чином, впровадження електронного журналу обліку успішності та відвідуваності учнів/студентів дозволить покращити якість освітянських послуг та змінімізує корупційну складову в освіті.

ПЕРЕЛІК ПОСИЛАНЬ

1. Hellas A. et. al. Predicting academic performance: a systematic literature review. In Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018 Companion). ACM, New York, NY, USA, 2018. Pp. 175–199. doi: https://doi.org/10.1145/3293881.3295783 .

2. World Economic Forum Global Competitiveness Report. 2011. URL: http://www3.weforum.org/docs/WEF_GCR_Report_2011-12.pdf .

3. Державна служба статистики України. [Електронний ресурс] – Режим доступу: http://www.ukrstat.gov.ua/ .

4. Сікорський П. І. Дидактичні поняття кредиту і модуля в контексті Болонського процесу // Шлях освіти. – 2004. – №2. – с. 19.

5. Методичні вказівки до виконання магістерської роботи освітнього рівня ―магістр‖ студентами усіх форм навчання для напряму підготовки 121 – ―Інженерія програмного забезпечення‖ / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 26 с.

6. What is Agile? – Неприбуткова організація Agile Alliancею [Електронний ресурс] – Режим доступу: https://www.agilealliance.org/agile101/.

7. Benington, Herbert D. (1 October 1983). «Production of Large Computer Programs» . IEEE Annals of the History of Computing. IEEE Educational Activities Department. 5 (4): 350–361.

8. Кент Бек. Экстремальное программирование. — СПб : Питер, 2002. — ISBN 5-94723-032-1.

9. Scrum (software development) – Комп’ютерний науковий портал GeeksforGeeks. [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/scrum-software-development/

10. Generics в Java 1.5, Kobylansky Stanislav, RSDN Magazine #1-2005. [Електронний ресурс] – Режим доступу: http://rsdn.org/article/patterns/generic-mvc.xml

11. Java Standard Edition 8, специфікація API. [Електронний ресурс] – Режим доступу: https://docs.oracle.com/javase/8/docs/api/ .

12. Why Is C# Among The Most Popular Programming Languages in The World? – Онлайн журнал Medium. [Електронний ресурс] – Режим доступу: https://medium.com/sololearn/why-is-c-among-the-most-popular-programming-languages-in-the-world-ccf26824ffcb .

13. Alexandrescu, Andrei (2001). Modern C++ Design: Generic Programming and Design Patterns Applied. Addison-Wesley. ISBN 0-201-70431-5.

14. Choosing your Java IDE – Онлайн-ресурс JavaWorld from IDG. [Електронний ресурс] – Режим доступу: https://www.javaworld.com/article/3114167/choosing-your-java-ide.html .

15. Пакет javax.swing – Офіційна документація Oracle. [Електронний ресурс] – Режим доступу: https://docs.oracle.com/javase/8/docs/api/index.html?javax/swing/package-summary.html .

16. Vos, Johan; Gao, Weiqi; Chin, Stephen; Iverson, Dean; Weaver, James L. Pro JavaFX 8: A Definitive Guide to Building Desktop, Mobile, and Embedded Java Clients. Apress. p. 616. ISBN 978-1-4302-6574-0.

17. Date, Chris. Database in depth : relational theory for practitioners. O'Reilly. ISBN 0-596-10012-4.

18. Офіційна сторінка JDBC. [Електронний ресурс] – Режим доступу: https://www.oracle.com/technetwork/java/javase/jdbc/index.html .

19. Офіційна сторінка MySQL. [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/relnotes/mysql/8.0/en/ .

20. Офіційна сторінка Microsoft SQL Server. [Електронний ресурс] – Режим доступу: https://www.microsoft.com/en-us/sql-server/default.aspx .

21. Офіційна сторінка SQLite. [Електронний ресурс] – Режим доступу: https://www.sqlite.org/about.html .

22. Дизайн-патерни – просто, як двері. [Текст] Будай А. – С.24-26.

23. Системні вимоги Java 8. [Електронний ресурс] – Режим доступу: https://www.java.com/en/download/help/sysreq.xml .

24. Освоюємо Java/Встановлення і налаштування середовища розробки – Вікіпідручник. [Електронний ресурс] – Режим доступу: https://uk.wikibooks.org/ .

25. Форма №2 “Звіт про фінансові результати”: методика підготовки. [Електронний ресурс]. – Режим доступу: http://osvita.ua/vnz/reports/accountant/17368/ .

26. Brooks, Fred (1986). Kugler, H.J. (ed.). No Silver Bullet Essence and Accidents of Software Engineering. Information Processing '86. Elsevier Science Publishers B.V (North-Holland). ISBN 0-444-70077-3.

27. Методичні вказівки для виконання розділу дипломної роботи щодо техніко-економічного обґрунтування вибору проектного рішення розробки та оцінки якості програмного забезпечення / Упор. Петрик М.Р., Кінах Я.І., Головатий А.І., Рогатинська Л.Р. – Тернопіль: Вид-во ТНТУ ім. І. Пулюя. – 2013. – 34 с.

28. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/rada/show/va042282-99 .

29. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98. [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/rada/show/v0007282-98 .

30. Наказ Міністерство охорони здоров'я України «Про затвердження Порядку проведення медичних оглядів працівників певних категорій». [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0846-07 .

ДОДАТКИ

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра програмної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою

програмної інженерії

Петрик Михайло Романович

"___ "_____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської дипломної роботи

на тему: «Розробка інформаційно-електронної системи для контролю відвідуваності та успішності студентів «iJournal» на основі технології JavaFX»

Миколюку Юрійю Мироновичу <СПм-15-300> ТЗ

Керівник роботи:

к.ф.-м.н., доцент Бойко І. В.

"___" _____ 2019р.

_____

Виконавець:

студент групи СПм-61

Миколюк Юрій Миронович

"___" _____ 2019р.

_____

м. Тернопіль – 2019

ЗМІСТ

1 ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану на 2019 рік, та згідно наказу на виконання дипломної роботи студента-магістра.

Тема проекту: «Розробка інформаційно-електронної системи для контролю відвідуваності та успішності студентів «iJournal» на основі технології JavaFX».

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Українська система освіти реорганізовується та показує досить хороші успіхи. Але, на сьогодні, є багато освітянських аспектів, які потребують покращення. В основному це стосується того, що учні/студенти не володіють достатньою інформацією про власну успішність та свій рейтинг, що знижує мотивацією до навчання, яка необхідна для кращого засвоєння навчального матеріалу.

Викладачі та вчителі в сучасній системі освіти витрачають багато часу на заповнення різноманітних паперових документів, зокрема журналів, замість того, щоб більше часу приділяти процесу навчання. Поточна оцінка в журналі є важливим фактором успішності учня/студента. Підсумкова/модульна оцінка в освітніх закладах розглядається як істотний фактор прогнозу успішності знань у рамках річної/семестрової оцінки по предмету/дисципліні. Саме тому систематичне виставлення оцінок у журнали необхідно, але робота із паперовими носіями потребує набагато більше часу ніж ведення електронного журналу.

За результатами виконаної роботи необхідно отримати електронну систему для контролю відвідуваності та успішності студентів, яка сприятиме підвищенню ефективності та інтенсифікації учбового процесу.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Функціональні характеристики

Програмне забезпечення має виконувати наступні дії:

- мультиплатворменість (використання на комп’ютерах з операційними системами Windows чи MacOS);
- простота інтерфейсу;
- можливість використовувати на різних комп’ютерах;
- заповнення журналу проведеного уроку;
- перегляд журналу за попередні дні;
- перегляд статистики учня;
- редагування даних, записаних в програму;
- легке розгортання програмного продукту.

3.2 Склад та параметри технічних засобів

1) ПК або планшетний комп’ютер з 128 Мб оперативної пам‘яті, встановленою системою Windows XP, Vista, Seven, 8, 8.1, 10, MacOS, Linux. Не менше 150 Мб вільного місця на жорсткому диску. Процесор Pentium 2 266 МГц або потужніший.

2) наявність встановленої JDK version 8 або вище з можливістю розгорнути java-програми.

3.3 Інформаційна та програмна сполучність

Програмний продукт повинен коректно функціонувати в різних операційних системах (Linux, Windows, MacOS), на яких доступне для встановлення JRE 8. Розроблювана бібліотека класів повинна бути пристосована до використання у інформаційних системах та програмних засобах. Розробку виконувати з використанням бібліотек та технологій мови Java в середовищі програмування Eclipse з використанням JDK 8. У якості СУБД використовувати SQLite.

4. СТАДІЇ РОЗРОБКИ

В ходів реалізації роботи проект повинен пройти крізь наступні стадії розробки:

- аналіз предметної області;
- визначення вимог до програмної системи;
- проектування архітектури;
- реалізація класів;
- реалізація графічного інтерфейсу програми;
- тестування результатів розробки;
- оформлення супровідної документації;
- здача роботи.

5. ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для програмного продукту повинні бути розроблені наступні документи:

- Пояснювальна записка;
- Технічне завдання;
- Презентаційний матеріал;
- Додатки.

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Розроблений програмний продукт має виконувати всі вимоги, що складаються з перерахованих у п. 3.1 характеристик.

Приймання проводиться спеціально створеною екзаменаційною комісією в термін до:

“__” _____ 201_р.

ДОДАТОК Б

Текст тези

УДК 004.414.38

Ю. Миколюк, І. Бойко, канд. фіз.-мат. наук, доц.
Тернопільський національний технічний університет ім. І. Пулюя, Україна

РОЗРОБКА ІНФОРМАЦІЙНО-ЕЛЕКТРОННОЇ СИСТЕМИ ДЛЯ КОНТРОЛЮ ВІДВІДУВАНОСТІ ТА УСПІШНОСТІ СТУДЕНТІВ

Y. Mykoliuk, I. Boyko, Ph.D, Assoc. Prof

DEVELOPMENT OF INFORMATION-ELECTRONIC SYSTEM FOR CONTROL OF VISITING AND SUCCESS OF STUDENTS

Українська система освіти реорганізовується та показує досить хороші успіхи. Але, на сьогодні, є багато освітянських аспектів, які потребують покращення. В основному це стосується того, що учні/студенти не володіють достатньою інформацією про власну успішність та свій рейтинг, що знижує мотивацією до навчання, яка необхідна для кращого засвоєння навчального матеріалу.

Викладачі та вчителі в сучасній системі освіти витрачають багато часу на заповнення різноманітних паперових документів, зокрема журналів, замість того, щоб більше часу приділяти процесу навчання. Поточна оцінка в журналі є важливим фактором успішності учня/студента [1]. Підсумкова/ модульна оцінка в освітніх закладах розглядається як істотний фактор прогнозу успішності знань у рамках річної/семестрової оцінки по предмету/дисципліні. [2]. Саме тому систематичне виставлення оцінок у журнали необхідно, але робота із паперовими носіями потребує набагато більше часу ніж ведення електронного журналу.

Впровадження системи державних стандартів освіти також додає навантаження на працівників системи освіти, що підштовхує освітян до пошуку нових методів оптимізації роботи у навчальних закладах.

На сьогоднішній день в Україні є велика кількість закладів освіти, мета яких – якісно та в повній мірі надати учням/студентам освітній рівень, відповідно до рівня навчального закладу. Станом на 2018 р. в Україні нараховується більше 15 тисяч закладів загальної середньої освіти та більше 600 закладів вищої освіти [3] (див. табл. 1).

Таблиця 1

Статистика навчальних закладів України станом на 2018 р.

Тип закладу	Кількість закладів	Кількість студентів/учнів	Кількість викладачів/вчителів
Заклад вищої освіти	652	1 522 250	152 977
Заклад загальної середньої освіти	15 521	4 041 652	441 394
Разом	16 173	5 563 902	594 371

Україна, як і багато інших розвинених держав використовує модульну систему освіти. Модуль – це логічно завершена система теоретичних знань та фактичних умінь з даної навчальної дисципліни, адаптованих до індивідуальних особливостей суб'єктів учіння з визначеним оптимальним часом на організацію її засвоєння [4]. Розбиття навчального матеріалу на модулі покращує сприйняття нового матеріалу студентами. Введення модульної системи також робить систему оцінювання знань більш прозорою. Але, через це викладачам необхідно проводити розрахунок успішності та відвідуваності кожного студента в кінці кожного модуля, що може займати багато часу.

Як варіант вирішення згаданих проблем системи освіти можна розглянути впровадження електронного журналу обліку успішності та відвідуваності студентів. Використання такого журналу дозволить зекономити багато часу викладачів, адже їм необхідно буде витрачати менше часу на аналіз успішності студентів у кінці модуля.

Вчителі зможуть якісніше використовувати цей час для того, щоб краще підготувати навчальний матеріал, ретельніше перевіряти роботи учнів/студентів, шукати індивідуальний підхід до кожного учня/студентів та багато іншого. А це, у свою чергу, значно поліпшить рівень наданих освітянських послуг та підвищить якість знань із предмету/дисципліни.

Іншими словами, перехід із паперового на електронний журнал обліку відвідуваності та успішності учнів/студентів надає багато переваг, а саме:

- раціональне використання часу вчителів/викладачів, адже вони матимуть можливість більше часу витрачати на навчання учнів/студентів тому, що зекономиться час котрий тратився на заповнення паперового журналу;
- зниження корупційної складової, адже оцінки будуть виставлятися он-лайн та мінімізується можливість помилки при ручному обчисленні середньої оцінки за модуль та кількості відвідувань завдяки автоматичному підрахунку;
- ширший функціонал електронного журналу, в порівнянні з паперовим, що забезпечить якісніший контроль успішності для самих учнів/студентів та посилить здорову конкуренцію серед них.

Перехід навчальних закладів з паперового журналу на електронний дозволить використовувати велику кількість додаткових функцій журналу. Наприклад, електронний журнал обліку успішності та відвідуваності студентів може мати функцію перевірку поточної загальної успішності та відвідуваності студента за певний період, а не лише в кінці модуля. А це допоможе студенту аналізувати свою продуктивність по окремій дисципліні. Також ця функція може допомогти батькам учнів проводити нагляд за успішністю та відвідуваністю своїх дітей-учнів.

Можливість генерувати графіки, що відображають успішність та відвідуваність учнів/студентів дозволить наочно представляти ефективність кожного.

Зберігання даних журналу в електронному вигляді, а не в паперовому дозволить зберігати дані в мережі Інтернет, що дасть можливість вчителю/викладачу проводити роботу з журналом без прив'язки до місця роботи та паперової версії журналу. Також це дозволить учням/студентам та їх батькам переглядати навчальну успішність та відвідуваність уроків дистанційно, навіть зі свого дому.

Таким чином, впровадження електронного журналу обліку успішності та відвідуваності учнів/студентів дозволить покращити якість освітянських послуг та змінімізує корупційну складову в освіті.

Література

[1] – Hellas A. et. al. Predicting academic performance: a systematic literature review. In Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018 Companion). ACM, New York, NY, USA, 2018. Pp. 175–199. doi: https://doi.org/10.1145/3293881.3295783 .

[2] – World Economic Forum Global Competitiveness Report. 2011. URL: http://www3.weforum.org/docs/WEF_GCR_Report_2011-12.pdf (дата звернення 30.05.2019).

[3] – Державна служба статистики України. [Електронний ресурс] – Режим доступу: http://www.ukrstat.gov.ua/ .

[4] – Сікорський П. І. Дидактичні поняття кредиту і модуля в контексті Болонського процесу // Шлях освіти. – 2004. – №2. – с. 19.

ДОДАТОК В

Диск

ДОДАТОК Г

UML діаграма класів