ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕХНОЛОГІЧНИЙ ФАХОВИЙ КОЛЕДЖ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Відділення комп'ютерних технологій

Циклова комісія інформаційно-комп'ютерної підготовки

Пояснювальна записка

до дипломного проекту
молодшого спеціаліста

на тему **" Програма для обліку дорогоцінного каміння з можливістю створення виробів"**

Виконав студент IV курсу
групи КІ-42
спеціальності 123 "Комп'ютерна інженерія"
Драбський Маркіян Володимирович

Керівник Почтарук Микола Миколайович

Рецензент _____

(прізвище та ініціали)

Львів – 2024 рік

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕХНОЛОГІЧНИЙ ФАХОВИЙ КОЛЕДЖ
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Відділення	**комп'ютерних технологій**
Циклова комісія	**інформаційно-комп'ютерної підготовки**
Освітньо-кваліфікаційний рівень	**"фаховий молодший бакалавр"**
Спеціальність	**123 "Комп'ютерна інженерія"**

ЗАТВЕРДЖУЮ
Заступник директора
з навчальної роботи
_____ М.В. Підвальний
" 18 " березня 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ ГРУПИ КІ-42

Драбському Маркіяну Володимировичу

1. Тема проекту **" Програма для обліку дорогоцінного каміння з можливістю створення виробів",**
керівник проекту **Почтарук Микола Миколайович,**
затверджені наказом від 15 березня 2024 року № 22-КС

2. Термін подання студентом проекту – до 10 червня 2024 року

3. Вихідні дані до проекту

Програма повинна забезпечувати:
- можливість внесення даних про дорогоцінне каміння до системи;
- редагування вже внесених даних;
- можливість сортування каміння за їх типом та ознакою прозорості;
- можливість створення виробів (намист);
- каміння в намисті сортувати за типом, якщо типи співпадають - за вартістю, а якщо співпадає вартість - за назвою.

4. Зміст розрахунково-пояснювальної записки

Титульний аркуш
Завдання на дипломне проектування
Анотація
Annotation
Перелік скорочень, символів і спеціальних термінів
Зміст
Вступ
Розділ 1 Аналіз технічного завдання
- 1.1 Аналітичний огляд
- 1.2 Обґрунтування поставленої задачі
- 1.3 Переваги обраного засобу розробки

Розділ 2 Проектування програми
- 2.1 Опис архітектури програми.
- 2.2 Опис схеми роботи програми.

Розділ 3 Розробка програми
- 3.1 Розробка графічного інтерфейсу.
- 3.2 Розробка бази даних.
- 3.3 Розробка програми.

Розділ 4 Експериментальна частина
4.1 Створення автоматичних тестів для програми
4.2 Мануальне тестування
4.3 Аналіз результатів
Розділ 5 Економічна частина
Розділ 6 Охорона праці
Висновки
Список літератури
Додаток

5. Перелік графічного матеріалу

Аркуш 1. Схема роботи програми.
Аркуш 2. Вигляд інтерфейсу програми.
Аркуш 3. Схема бази даних.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічна частина	Комар І.О.		
Охорона праці	Романко С.М.		

7. Дата видачі завдання – 19 березня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Збір та опрацювання матеріалу за темою дипломного проекту	22.04.24 - 02.05.24	
2	Проектування програми	03.05.24 - 07.05.24	
3	Розробка програми	08.05.24 - 15.05.24	
4	Експериментальна частина	16.05.24 - 20.05.24	
5	Розробка розділу "Економічна частина"	21.05.24 - 25.05.24	
6	Розробка розділу "Охорона праці"	26.05.24 - 29.05.24	
7	Розробка графічного матеріалу	30.05.24 - 03.06.24	
8	Розробка та оформлення пояснювальної записки	04.06.24 - 07.06.24	
9	Перевірка на плагіат	08.06.24 - 09.06.24	

Студент _____ **Драбський М.В.**

Керівник проекту _____ **Почтарук М.М.**

Завдання на дипломний проект розглянуте і схвалене на засіданні циклової комісії інформаційно-комп'ютерної підготовки

Протокол № 7 від 14 березня 2024 р.

Голова циклової комісії _____ / Ковтонюк О.Г. /

АНОТАЦІЯ

Тема роботи є актуальною та має практичне значення у зв’язку і є завершеним самостійним дослідженням. Програмний продукт є системою, що дозволяє полегшити роботу з сортування та формування виробів в ювелірній сфері. Систему створено за допомогою мови Java. Може використовувати базу даних MySql або H2.

Об’єкт дослідження – Збереження та обробка інформації про дорогоцінні камені.

Предмет дослідження – Система, що дозволить зберігати та обробляти інформацію про дорогоцінні камені.

Методи дослідження - розробка системи, що дозволить зберігати та обробляти інформацію про дорогоцінні камені.

Метою роботи було розробити та впровадити систему, що дозволить зберігати та обробляти інформацію про дорогоцінні камені.

Структура дипломної роботи складається з анотації, вступу чотирьох основних розділів, обґрунтування економічної частини, охорона праці, висновків та переліку посилань.

Повний обсяг роботи складає 68 аркушів, вона містить 3 додатки та перелік посилань на використані джерела з 20 найменувань. У роботі наведено 27 рисунків та 6 таблиць.

Ключові слова: WINDOWS, JAVA, INTELLIJ IDEA, MAVEN, MySQL.

ANNOTATION

The topic is relevant and of practical importance in connection with and is a completed independent study. The software product is a system that facilitates the work of sorting and forming products in the jewelry industry. The system was created using the Java language. It can use MySql or H2 database.

Object of research - Storage and processing of information about gemstones.

Subject of research - A system that will allow to store and process information about gemstones.

Research methods - development of a system that will allow storing and processing information about gemstones.

The purpose of the work was to develop and implement a system that would allow storing and processing information about gemstones.

The structure of the thesis consists of an abstract, introduction of four main chapters, justification of the economic part, labor protection, conclusions and a list of references.

The total volume of the work is 68 pages, it contains 3 appendices and a list of references to the sources used from 20 titles. The work contains 27 figures and 6 tables.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І СПЕЦІАЛЬНИХ ТЕРМІНІВ

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

Maven – це засіб автоматизації роботи з програмними проєктами, який спочатку використовувався для Java проєктів;

JDK (Java Development Kit) – безкоштовний розповсюджуваний Oracle комплект розробника застосунків на мові Java, який включає до себе компілятор Java, стандартні бібліотеки класів Java;

JVM (Віртуальна машина Java) – набір комп'ютерних програм та структур даних, що використовують модель віртуальної машини для виконання інших комп'ютерних програм чи скриптів;

IDE – Інтегроване середовище розробки;

Java – об'єктно-орієнтована мова програмування;

JavaFx – бібліотека для створення зовнішнього викладу програми

Junit – бібліотека для тестування.

Spring framework – великий java фреймворк з великою кількістю модулів заточених під певні завдання.

Зміст

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І СПЕЦІАЛЬНИХ ТЕРМІНІВ..... 6
ВСТУП..... 8
РОЗДІЛ 1АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ..... 9
1.1.Аналітичний огляд..... 9
1.2.Обґрунтування поставленої задачі..... 9
1.3.Переваги обраного засобу розробки..... 10
2.1. Опис архітектури програми...... 13
2.2. Опис схеми роботи програми...... 14
РОЗДІЛ 3 РОЗРОБКА ПРОГРАМИ..... 16
3.1. Розробка графічного інтерфейсу...... 16
3.2. Розробка бази даних..... 21
3.3. Розробка програми..... 23
РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА..... 27
4.1. Створення автоматичних тестів для програми...... 27
4.2. Мануальне тестування...... 28
4.3. Аналіз результатів...... 35
РОЗДІЛ 5 ЕКОНОМІЧНА ЧАСТИНА..... 36
5.1.ВИБІР ТА ОБҐРУНТУВАННЯ БАЗОВОГО ПРОГРАМНОГО ПРОДУКТУ 36
Технічні та економічні показники аналогічних програмних продуктів..... 36
5.2 Розрахунок собівартості нового програмного продукту..... 39
5.3 Визначення ціни програмного продукту..... 46
РОЗДІЛ 6 ОХОРОНА ПРАЦІ..... 47
СПИСОК ЛІТЕРАТУРИ..... 58
ДОДАТКИ..... 60

ВСТУП

Дорогоцінне каміння завжди привертало увагу людей своєю красою, рідкісністю та вартістю. Від давніх часів люди використовували дорогоцінні камені для прикрас, обрядів та символіки. Сьогодні, в епоху технологій, облік та управління дорогоцінними каменями стає все важливішим завданням.

Розробка програми для обліку та роботи з дорогоцінним камінням має великий потенціал для бізнесу та майстерень. Ця програма може спростити процес ведення обліку, забезпечити точність даних та допомогти в управлінні запасами. Крім того, можливість створення виробів з дорогоцінного каміння додає цій темі ще більше цікавості та практичності.

Поставлена задача покликана спростити проблему організації реалізації продукції, її впорядкування, та менеджменту.

РОЗДІЛ 1АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1. Аналітичний огляд

Оглянемо поставлене завдання:

Програма повинна забезпечувати:

- можливість внесення даних про дорогоцінне каміння до системи;
- редагування вже внесених даних;
- можливість сортування каміння за їх типом та ознакою прозорості;
- можливість створення виробів (намист);
- каміння в намисті сортувати за типом, якщо типи співпадають - за вартістю, а якщо співпадає вартість - за назвою.

З нього можна зробити висновок, що програма орієнтована на працівників ювелірної сфери.

Система повинна зберігати 2 типи сутностей: дорогоцінні камені та вироби (намиста). Також для них потрібно забезпечити CRUD (create, read, update, delete) операції. Потрібно додати підтримку відображення інформації про вагу каменів у виробі та їх ціну. Також потрібно забезпечити можливість сортування каміння за ознакою прозорості у намисті. Можливість сортування також повинна бути реалізована. Порядок пріоритетності параметрів такий: тип > вартість > назва.

1.2. Обґрунтування поставленої задачі

Провівши дослідження, можна сказати, що спеціалізованих програм для обробки інформації у ювелірній сфері дуже мало. Цей факт дає можливість сподіватися на успіх проекту. Хоч ринок користувачів не є великим, проте працівники цієї сфери зможуть дозволити отримати спеціалізовану систему, якщо це дозволить наростити ефективність та організацію даних.

Цільовою аудиторією цього продукту є невеликі майстерні, що проводили облік та обробку інформації про дорогоцінне каміння на папері, або ж ті, що робили це у непрофільних додатках. Потенційно цей продукт може бути зручнішим і для великих підприємств, якщо продовжити його розвиток під конкретного замовника.

Враховуючи ці обставини вважаю, що ця тема може зайняти свою нішу.

1.3.Переваги обраного засобу розробки

Оскільки потенційними користувачами системи є майстерні, важливою властивістю для продукту є можливість роботи на будь-якому обладнанні. Ця властивість робить Java ідеальною мовою програмування для даного проекту. Завдяки віртуальній машині JVM (Java Virtual Machine), Java-код може запускатися на будь-якій платформі, що має JVM, незалежно від операційної системи або архітектури процесора. Це відповідає відомому слогану Java: "Напиши один раз, запускай будь-де" ("Write once, run anywhere").

Також з плюсів даної мови можна виділити:

- Бібліотеки
- Спільнота
- Простота

Для розробки також використовуватиметься Spring Framework.

Spring Framework - це популярний фреймворк з відкритим кодом для Java, який використовується для спрощення розробки Java-додатків. Він пропонує широкий спектр модулів, які допомагають розробникам вирішувати різні завдання, такі як:

- **Веб-розробка:** Spring MVC - це фреймворк MVC (Model-View-Controller), який використовується для створення веб-додатків Java. Він надає зручний інтерфейс для розробки контролерів, обробки HTTP-запитів та відповідей, а також підтримку JSON.
- **Робота з базами даних:** Spring Data JPA - це фреймворк для об'єктно-реляційного відображення (ORM), який спрощує доступ до баз даних з Java-додатків. Він дозволяє розробникам працювати з об'єктами Java замість складних SQL-запитів.
- **Аспектно-орієнтоване програмування (AOP):** Spring AOP дозволяє розробникам реалізовувати поперечні аспекти, такі як безпека, ведення журналів та транзакції, без шкоди для основного коду. Це робить код більш модульним та легким у підтримці.

Spring базується на використанні **dependency injection**.

Dependency Injection (DI) - це принцип програмного забезпечення, який описує спосіб надання залежностей компонентам програми. Замість того, щоб компонент сам створював свої залежності, вони надаються йому ззовні. Це робить код більш модульним, тестованим та легким у підтримці.

Тепер розглянемо бібліотеки для створення графічного інтерфейсу:

1. **JavaFX**:

Переваги:

- Сучасна та потужна бібліотека з широким набором функцій
- Декларативний стиль програмування, що робить код простим та читабельним
- Підтримка кроссплатформності, що дозволяє запускати програми на різних операційних системах
- Активна спільнота та багато ресурсів для навчання

Недоліки:

- Може бути складною для початківців
- Деякі функції можуть бути не такими стабільними

2. **Swing**:

Переваги:

- Зріла та стабільна бібліотека з багаторічною історією
- Широко використовується та має велику спільноту
- Проста у вивченні та використанні

Недоліки:

- Не така сучасна, як JavaFX
- Може бути громіздкою та складною для створення складних інтерфейсів
- Не підтримує декларативний стиль програмування

3. **SWT**:

Переваги:

- Нативна бібліотека, що забезпечує високу продуктивність
- Кроссплатформна та підтримує широкий спектр операційних систем
- Проста у вивченні та використанні

Недоліки:

- Не така поширена, як Swing або JavaFX
- Може мати меншу спільноту та менше ресурсів для навчання
- Деякі функції можуть бути не такими потужними, як у Swing або JavaFX

4. **AWT**:

Переваги:

- Найстаріша та найпростіша бібліотека GUI в Java
- Проста у вивченні та використанні

Недоліки:

- Застаріла та не рекомендується для нових проектів
- Не така потужна або гнучка, як Swing, JavaFX або SWT
- Може мати проблеми з сумісністю з сучасними операційними системами

Отже, оскільки для нас важлива така характеристика як кроссплатформність, найкращим вибором буде використовувати бібліотеку JavaFX. Також стане у пригоді декларативний стиль програмування з використанням цієї бібліотеки та активна спільнота.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ПРОГРАМИ.

2.1. Опис архітектури програми.

З моменту свого виникнення область комп'ютерних наук стикалася з проблемами, пов'язаними зі складністю програмних систем. Раніше розробники вирішували ці проблеми шляхом обрання відповідних структур даних, розробки алгоритмів та застосування концепції розмежування повноважень.

Один з ключових етапів у реалізації мобільного додатку - це розробка архітектури. Якщо архітектура побудована належним чином, вона сприяє майбутньому розширенню та розвитку програми.

Архітектура програмного додатку - це структура, яка включає програмні компоненти, їх видимі властивості та відносини між ними.

Основна ідея архітектури полягає в зниженні складності систем шляхом абстракції та розмежування повноважень.

Розробка системи за темою "Програма для обліку дорогоцінного каміння з можливістю створення виробів" з використанням JavaFx та Spring boot проводитиметься за архітектурою MVC, що розшифровується як Model (модель), Viev (представлення), Controller (контролер).

Модель у цьому випадку - це центральний компонент системи, який безпосередньо управляє даними, логікою та правилами програми. У цій системі, модель буде включати структури даних для дорогоцінного каміння, їх характеристики, а також інформацію про вироби, які можна створити з цих каменів.

Представлення відповідає за відображення даних, отриманих від моделі, користувачеві. Він представляє модель у формі, яка підходить для взаємодії, у цій системі - графічний інтерфейс користувача, створений за допомогою JavaFX.

Контролер пов'язує попередні 2 типи елементів. Отримавши інформацію від моделі, контроллер обробляє її щоб розмістити на представленні.

2.2. Опис схеми роботи програми.

Програма складатиметься з 2х основних блоків:

- Робота з камінням
- Робота з намистами

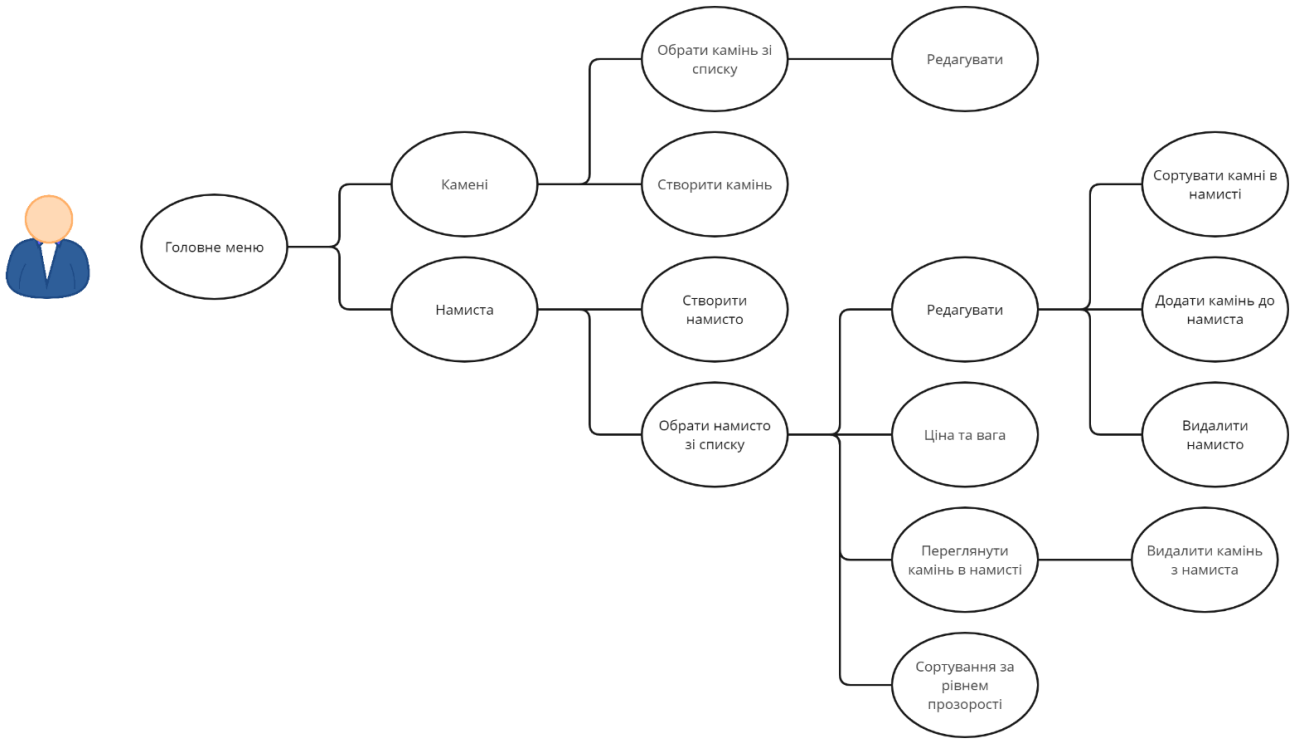

Рис. 2.2.1 – Схема роботи програми.

У першому блоці необхідно реалізувати можливість обрати камінь за назвою, або створити новий. Для каменю визначено такі властивості:

- Ім’я
- Вага
- Прозорість
- Ціна за карат
- Тип

Також потрібно додати можливість редагувати вже створені камені. Перейти до меню для цього можна з вікна конкретного каменю.

Переходимо до меню намист. Нам також потрібно створити меню вибору конкретного намиста з списку. Створення нового намиста потребує лише його імені. Щойно створене намисто завжди буде пустим. У вікні намиста потрібно

відображати ціну та вагу. Також важливо додати фільтрування каменів у намисті за зазначеною у завданні ознакою, а саме, рівнем прозорості. У цьому випадку використовуватиму 2 поля, що позначатимуть нижню та верхню межу. Для того, щоб додати камінь до намиста потрібно буде перейти до меню редагування намиста. Там нас зустріне перелік каменів, що не належать до жодного намиста. Також там знаходяться 2 кнопки:

- Для видалення намиста
- Для сортування намиста за відповідними параметрами.

При видаленні, усі камені, що знаходилися у намисті, стають доступними для додавання в нові. Для того, щоб видалити камінь з намиста слід повернутися до меню намиста, та обрати потрібний камінь. У цьому меню буде можливість видалити камінь.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМИ

3.1. Розробка графічного інтерфейсу.

Розмітка графічного інтерфейсу в JavaFx зберігається у файлах з розширенням fxml. Для розробки графічного інтерфейсу використовую середовище Gluone Scene Builder.

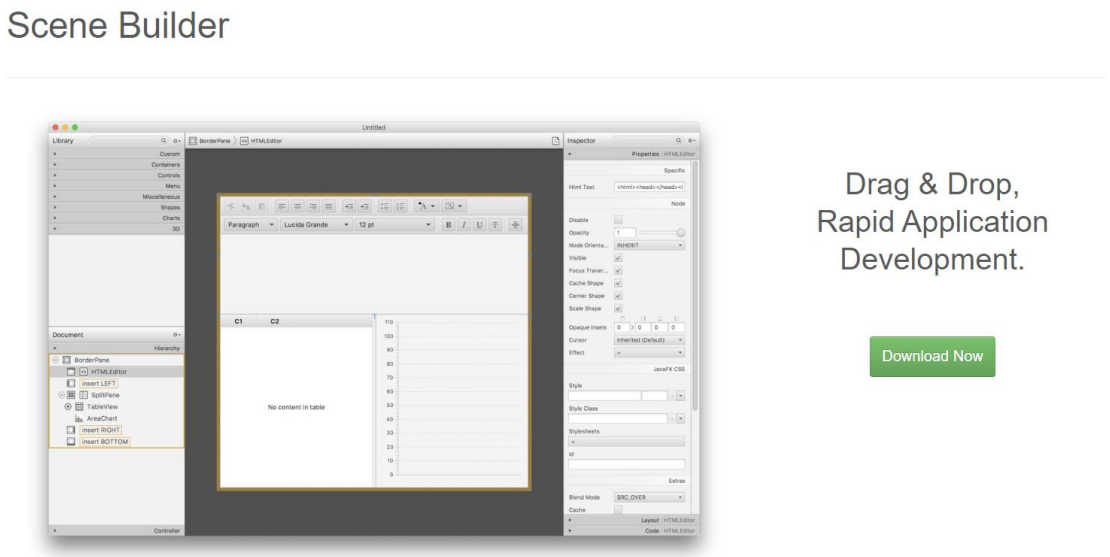

Рис. 3.1.1 – Вигляд сайту Gluone Scene Builder

Це візуальний редактор, що дозволяє перетягувати потрібні елементи на потрібне місце. Також Scene Builder може відтворити створений екран, що дає змогу оцінити як він виглядатиме в проекті.

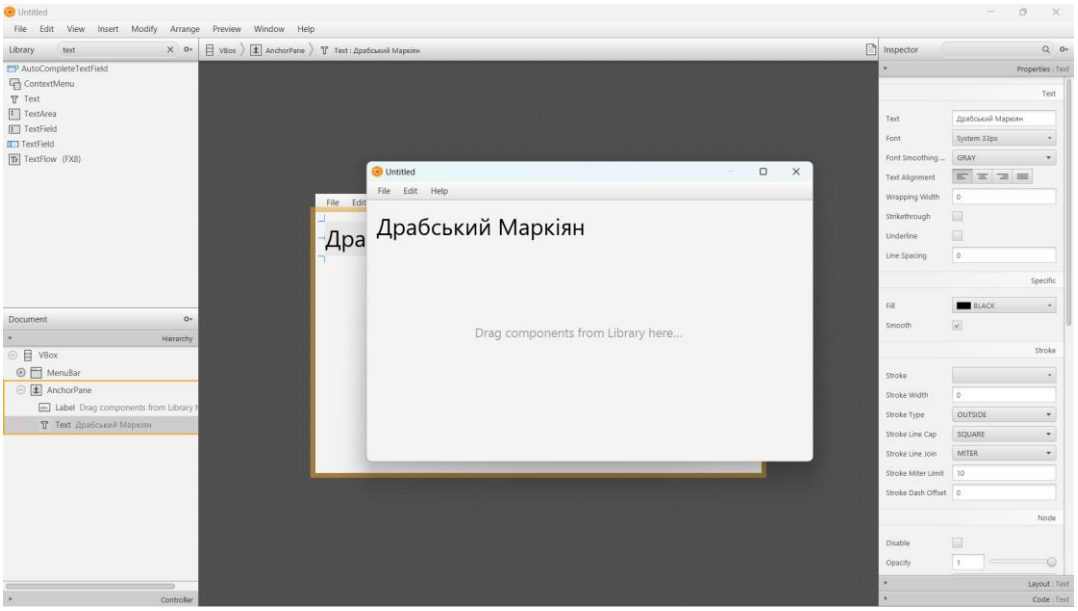

Рис. 3.1.2 – Вигляд середовища та демонстраційного вікна

З використанням цього інструменту розробив початкове меню. Воно складається з 2х секцій:

- Секція меню. На ній знаходиться назва програми, а також на інших вікнах за потреби додаватимуться кнопки дій.
- Робоча секція. На цьому вікні тут знаходиться тільки заклик обрати потрібний пункт меню, та 2 кнопки з навігацією до них.

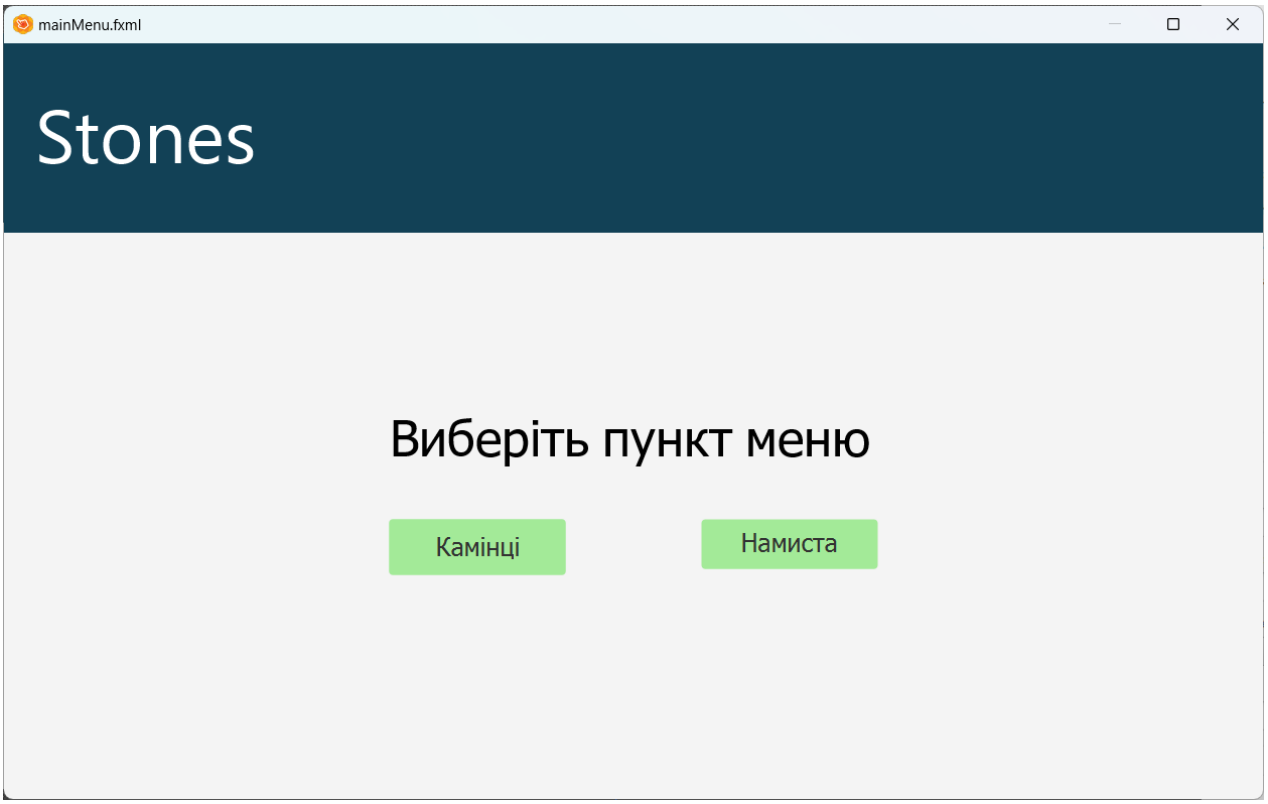

Рис. 3.1.3 – Вигляд початкового меню

Створив меню вибору камінців, воно застосовуватиметься у 3х випадках: у меню намист, у меню вибору камінців, та у меню додавання каменю в намисто.

На ньому розташовано 12 кнопок, кожна з яких вестиме до конкретного каменю/намиста. Якщо збережених елементів більше 12, стають активними кнопки керування сторінкою: кнопка назад, якщо сторінка не перша, та кнопка вперед, яко сторінка не остання. Також зверху присутні 2 кнопки, що матимуть різну назву залежно від сценарію, у якому використовуються, та мають викликати такі дії:

- Створення каменю
- Створення намиста
- Сортування каменів в намисті

- Видалення намиста

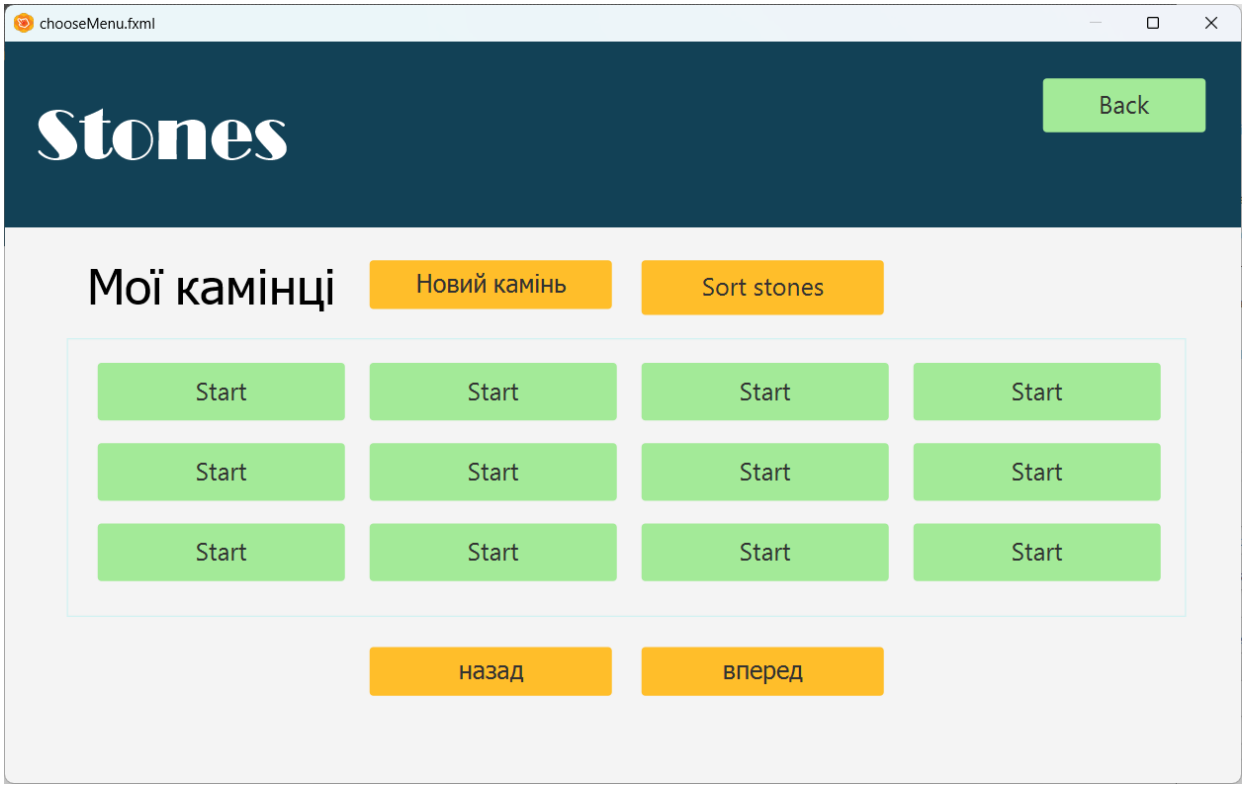

Рис. 3.1.4 – Вигляд меню вибору

Далі налаштував вікно створення каменю. Воно містить Місце для заповнення всіх необхідних даних, таких як:

- Ім’я
- Вагу
- Прозорість
- Ціну за карат
- Тип

Всі поля крім типу приймають текстову інформацію, що для числових даних буде змінено в коді. Тип ж є випадним списком з визначеними з бази даних типами.

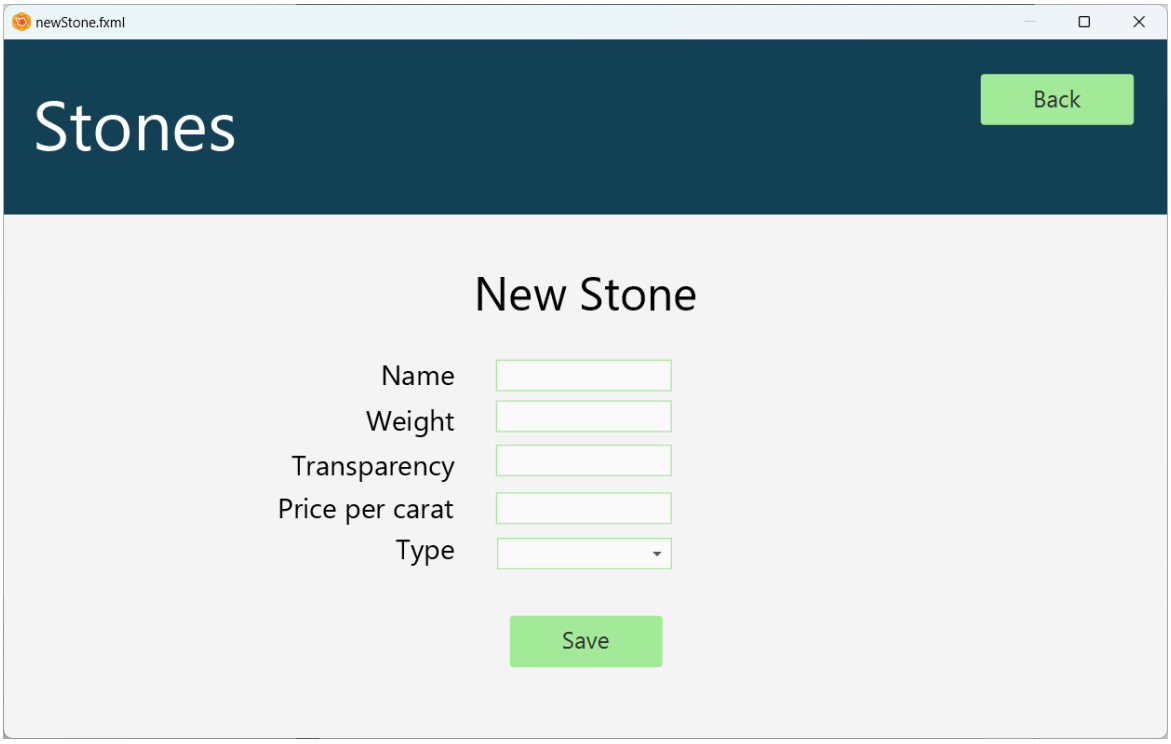

Рис. 3.1.5 – Вигляд меню створення каменю

Одразу ж створив вікно перегляду інформації про камінь. Воно дуже схоже до минулого, з такими відмінностями:

- Кнопка збереження перейменована в кнопку редагування
- Тип тепер також поле, не випадний список
- Всі поля є неактивними, тобто значення в них неможливо змінити

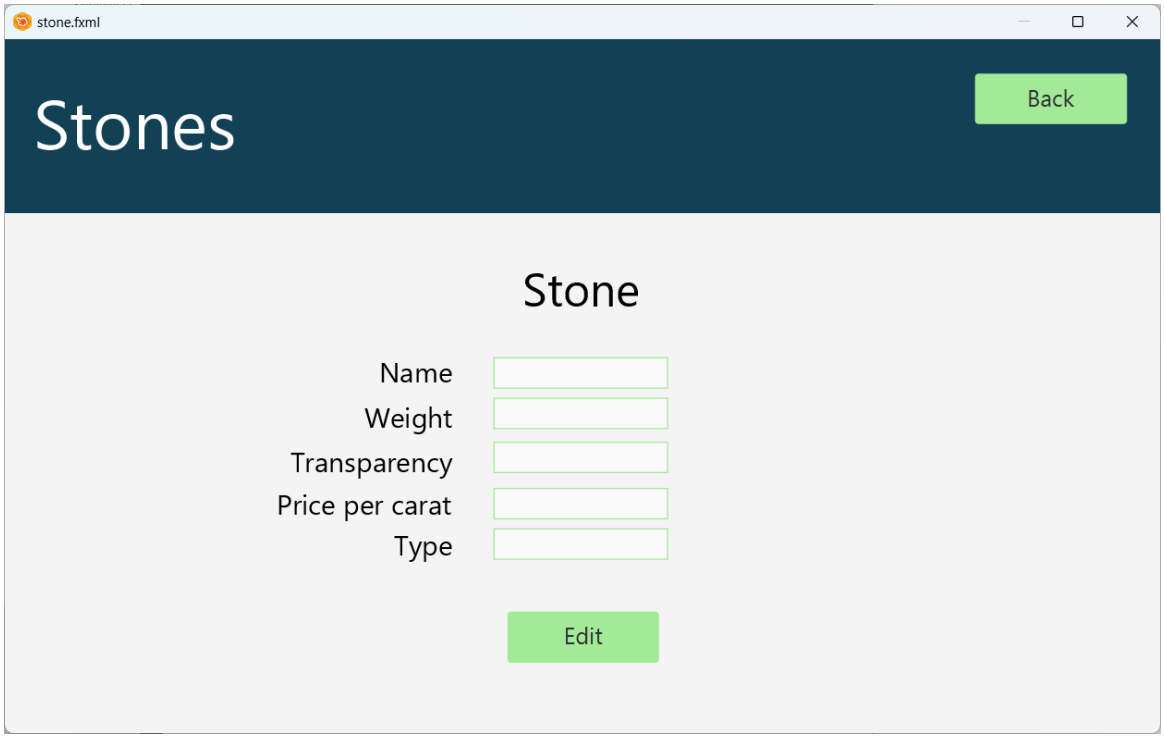

Рис. 3.1.6 – Вигляд меню огляду каменю

Також редагував це вікно для створення нового для створення намиста. У ньому залишив тільки поле імені, всі решта параметри розраховуватимуться автоматично.

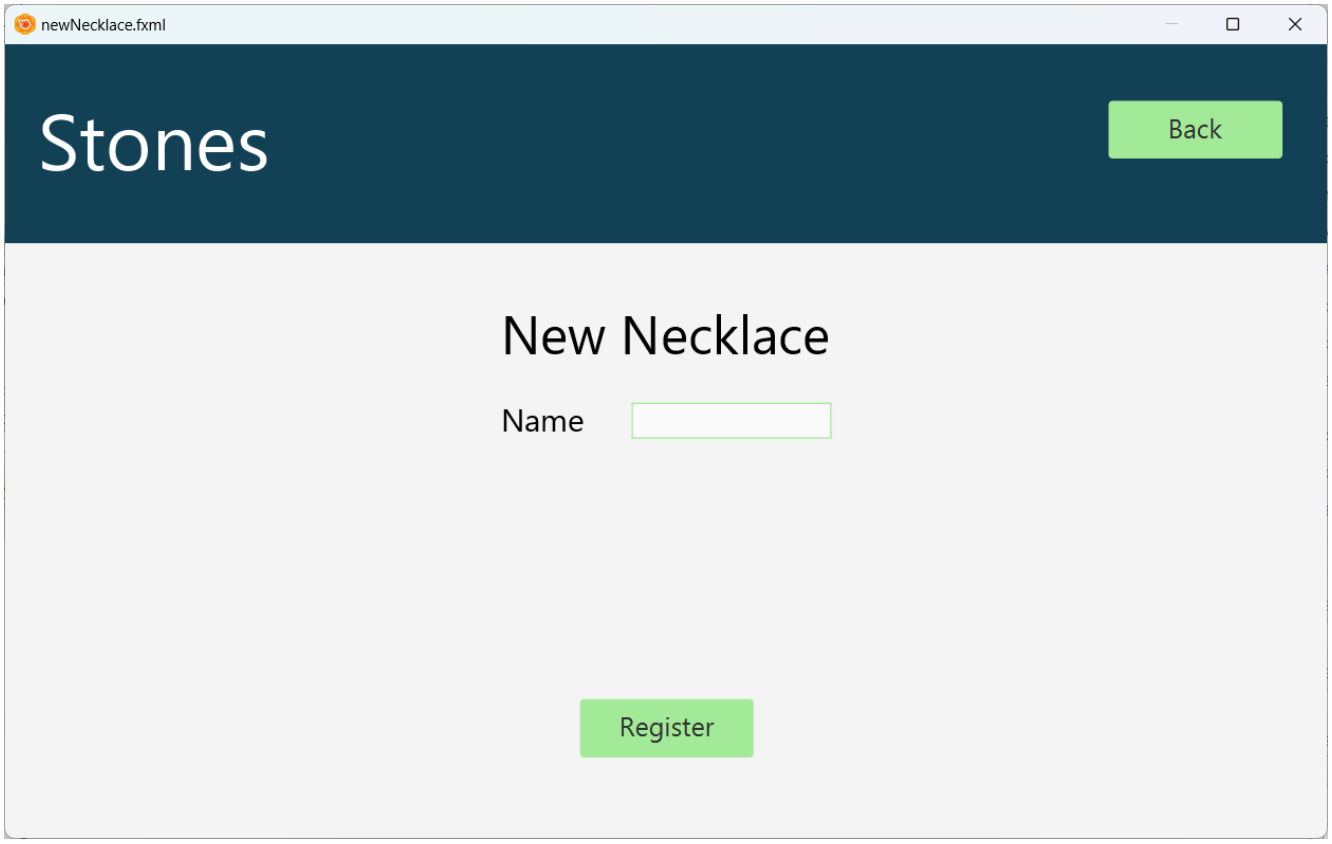

Рис. 3.1.7 – Вигляд меню створення намиста

Також створив вікно перегляду намиста, що містить кнопку його редагування. Поля для імені, вартості та ваги, а також поля для задання меж прозорості. Також має 4 кнопки для відображення каменів (залежно від типу будуть замінені відповідною картинкою), та кнопками навігації.

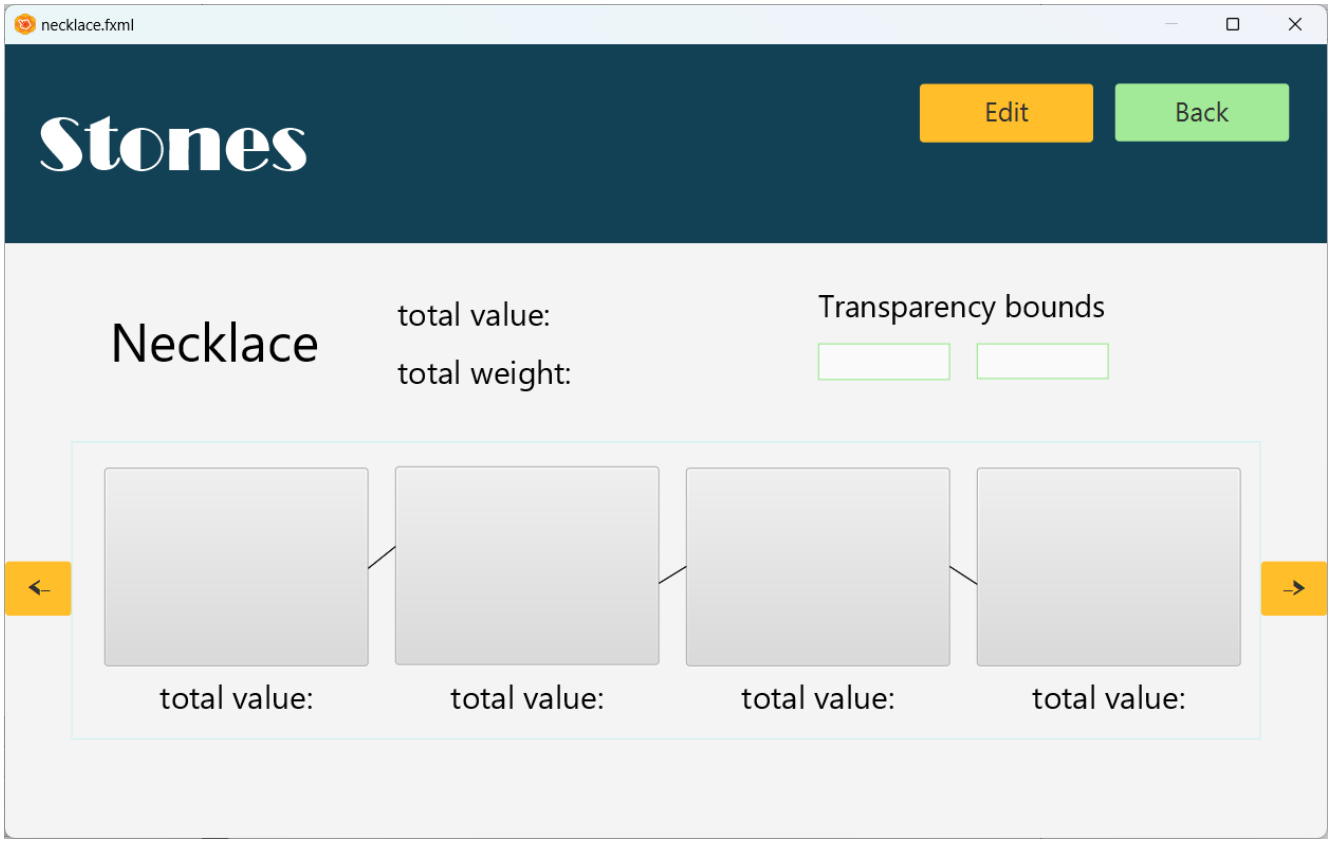

Рис. 3.1.8 – Вигляд меню намиста намиста

Додатково для виведення різних повідомлень створив вікно з текстовим полем для цього.

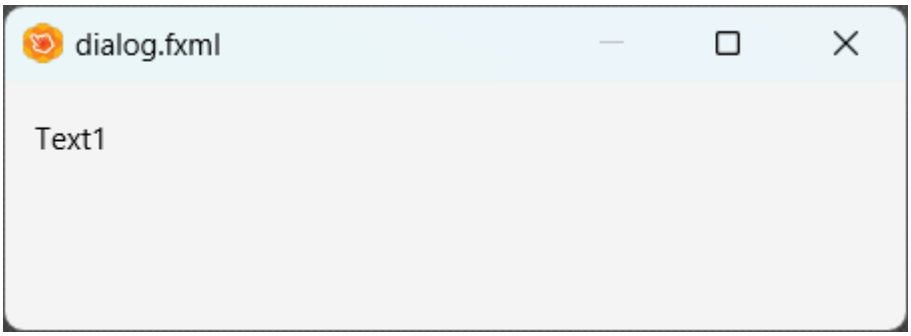

Рис. 3.1.9 – Вигляд діалогового меню

3.2. Розробка бази даних

Оскільки одною з вимог нашої системи є кросплатформеність, ми не повинні залежати від конкретної реалізації реляційної бази даних. Для цього передам процес створення таблиць, та встановлення зв'язку з базою даних модулю spring data. В такому разі необхідно буде створити класи entity для наших об'єктів в яких описати відношення між ними. Для кожного з них додатково використовуватиму

анотації бібліотеки Lombok, що дозволяє генерувати відповідні методи при компіляції.

Створив entity клас тип каменю, в якому описав всі потрібні поля а саме:

- Id
- Ім’я
- Вагу типу в рейтингу каменів
- Камені

Описав відношення між типами та каменями як один до багатьох, отже камінь може мати тільки один тип, тип ж може мати безліч каменів.

Створив entity клас намисто, в якому описав всі потрібні поля а саме:

- Id
- Ім’я
- Позицію наступного каменю
- Камені

Описав відношення між таблицями каменів та намист, як один до багатьох, отже камінь може бути тільки в одному намисті, намисто ж може мати безліч каменів.

Створив entity клас камінь, в якому описав всі потрібні поля а саме:

- Id
- Ім’я
- Ціну за карат
- Вагу
- Прозорість
- Позицію в намисті
- Тип каменю
- Та намисто, у якому може знаходитись

Описав відношення між таблицями каменів та намист, як один до багатьох, отже камінь може бути тільки в одному намисті, намисто ж може мати безліч каменів.

Для опису методів звернення до бази даних потрібно створити репозиторії. В цій системі всі вони наслідують інтерфейс JpaRepository, що має вже певну кількість стандартних методів, таких як збереження, отримання запису за id, пошук усіх, тощо. Для того, щоб створити додаткові методи, потрібно описати їх використовуючи спеціальний синтаксис, або якщо за допомогою цього важко, неможливо зробити це, можна використати анотацію @Query та передати sql запит написаний мовою jpql.

Створив репозиторій для типів каменю, у ньому не додавав додаткових методів, адже використовую тільки вже реалізовані.

Створив репозиторій для намист, у ньому додав 2 методи:

- Для отримання намиста за id
- Для отримання сторінки намист

Створив репозиторій для каменів, у ньому додав методи для:

- Отримання сторінки каменів
- Отримання сторінки каменів, які не є в намисті
- Отримання сторінки каменів що знаходяться в намисті, та мають зазначені параметри прозорості
- Отримання каменів в намисті

3.3. Розробка програми

Спершу створив новий проект, та додав необхідні залежності до файлу pom.xml, а саме:

- org.springframework.boot:spring-boot-starter-test
- org.springframework.boot:spring-boot-starter-data-jpa
- org.springframework.boot:spring-boot-starter-data-jdbc
- org.projectlombok:lombok
- com.mysql:mysql-connector-j
- com.h2database:h2
- org.openjfx:org.openjfx
- org.openjfx:javafx-controls

Та додаю батьківською конфігурацію spring-boot-starter-parent

Створив клас StonesApplicationNative, що слугуватиме вхідною точкою програми. Він наслідується від javafx Application, та перевизначає методи:

- init - запускаю spring context, та пов’язую його з FXMLLoader.
- start - запускаю першу сцену
- stop – завершую роботу spring context.

Та додаю main метод, що і запустить програму.

Переніс entity класи та репозитоії до цього проекту.

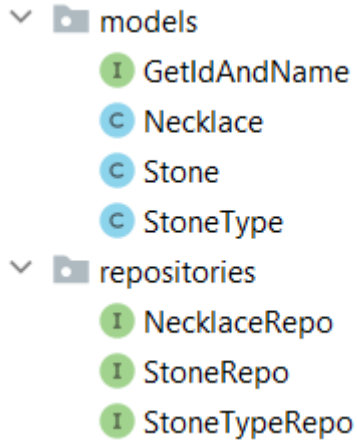

Рис. 3.3.1 – Моделі та репозиторії

Додав файли налаштування програми, перший з них: application.yaml. Містить загальні конфігурації, що стосуються програми загалом, а саме:

- Визначення активного профілю
- Визначення типу веб додатку

Також створив 2 файли налаштування для профілів баз даних mysql та h2. Вони містять інформацію про з’єднаяння з базою даних.

В такому випадку, якщо нам потрібно додати підтримку іншої реалізації реляційної бази даних, нам потрібно буде додати залежність для підключення, та додатковий файл налаштування.

Оскільки база даних h2, є in memory, тобто створюється в базі даних, додав файл data.sql, що заповнюватиме її тестовими даними.

Переходжу до розробки контролерів.

Для початку створив абстрактний клас SceneOpener, що містить:

- Сховище, об’єкт, що передається усім контролерам за допомогою dependency injection.
- Методи для відкриття певного вікна
- Прослуховувачі змін полів, що містять певні правила, наприклад, що у числові поля можна вводити тільки числа, у поле прозорості можна передати тільки значення від 0 до 1
- Метод відкриття діалового вікна

Розробляю реалізації контролерів. Початкові зразки класів отримав з scene builder:

Sample Skeleton for 'necklace.fxml' Controller Class

```
package com.example.diploma.controllers.mainMenu.necklaces.necklace;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.shape.Line;
import javafx.scene.text.Text;

public class NecklaceController {

    @FXML
    private Button backButton;

    @FXML
    private TextField bound1;

    @FXML
    private TextField bound2;

    @FXML
    private Line firstLine;

    @FXML
    private Button newStoneButton;

    @FXML
    private Button nextPageButton;

    @FXML
```

Copy Save as... Java Comments Full

Рис. 3.3.2 – Зразок контролера

Додав анотації, а також поля, значення який передаватимуться під час створення.

У методі initialize виконую налаштування вікна. У цьому випадку:

- Встановив стандартні значення для полів границь прозорості.
- Встановив слухачі зміни цих полів
- Отримую намисто з репозиторію, та встановлюю значення відповідних інформаційних полів.
- Встановив логіку для кнопок.

За таким самим принципом виконав налаштування інших контроллерів. У випадах, коли одна сцена відповідає за кілька сценаріїв, використовую switch для виклику актуального методу налаштування.

РОЗДІЛ 4
ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

4.1. Створення автоматичних тестів для програми.

Тестування – це процес перевірки продукту, як програмного, так і фізичного, на відповідність їх до зазначених характеристик.

Для тестування використовував бібліотеку org.testfx:testfx-junit5, а саме клас FXRobot, що дозволяє запрограмувати натискання кнопок графіччного інтерфейсу. За допомогою того ж dependency injection перевіряв як змінюються значення у потрібних полях програми, а за допомогою методу toNode витягував потрібні поля програми для перевірки правильності їх параметрів. Для тестування слід використовувати профіль h2 в файлі application.yaml.

- Розробив такі тестові класи:
- Клас MainMenuTest має у собі тести, що перевіряють чи правильне вікно відкрито після натискання кнопок переходу до меню каменів та меню намист.
- Клас AddingStoneToNecklaceTest як зрозуміло з назви містить тести на додавання каменю до намиста.
- Клас CreateNecklaceTest містить тести на створення намиста, та перевіряє випадки, коли ім’я є пробілом або пустим.
- Клас DeleteStoneFromNecklaceTest перевіряє чи правильно працює видалення каменю з намиста.
- Клас SortStonesInNecklaceTest перевіряє правильність сортування каменів в намисті після натискання потрібної кнопки.
- Клас CreateStoneTest пробує створити камінець заповнюючи поля різними значеннями та перевіряє реакцію програми.
- Клас EditStoneTest також пробує змінити камінь за допомогою внесення в відповідні поля певних змін, та перевіряє чи збережеться камінь.
- Клас RegularStoneViewTest перевіряє чи правильно заповнені поля каменю при перегляді його властивостей у меню каменів.

- Клас StoneInNecklaceTest перевіряє чи правильно заповнені поля каменю при перегляді його властивостей у меню намиста, та як налаштовано кнопки.

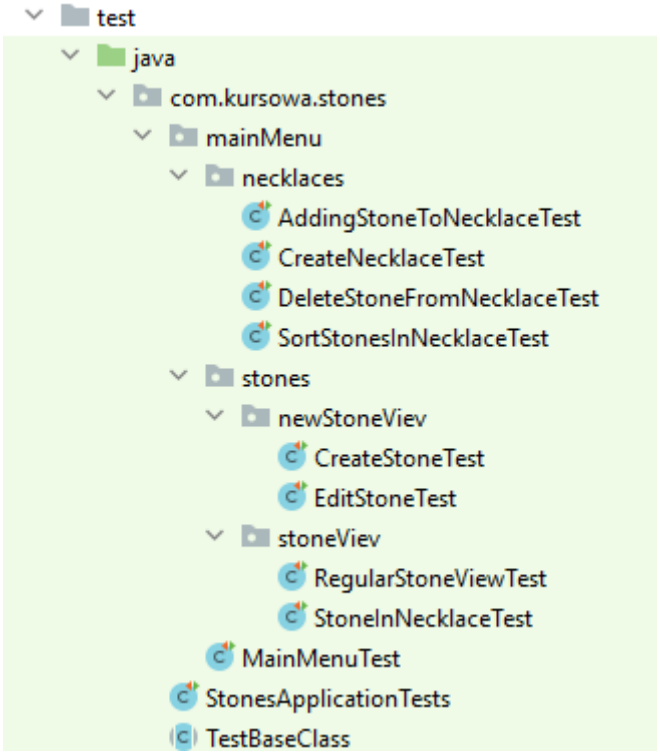

Рис. 4.1.1 – Автоматичні тести

4.2. Мануальне тестування.

Мануальне або ручне тестування — це процес ручної перевірки програмного забезпечення на помилки. Тестувальник має відігравати роль користувача програми й використовувати властивості програми для знаходження помилок у роботі програми.

Під час ручного тестування тестер перевіряє ключові функції програмного додатка, а аналітики виконують тестові випадки.

Проводжу тестування:

Обрав меню каменів та переглянув наявні.

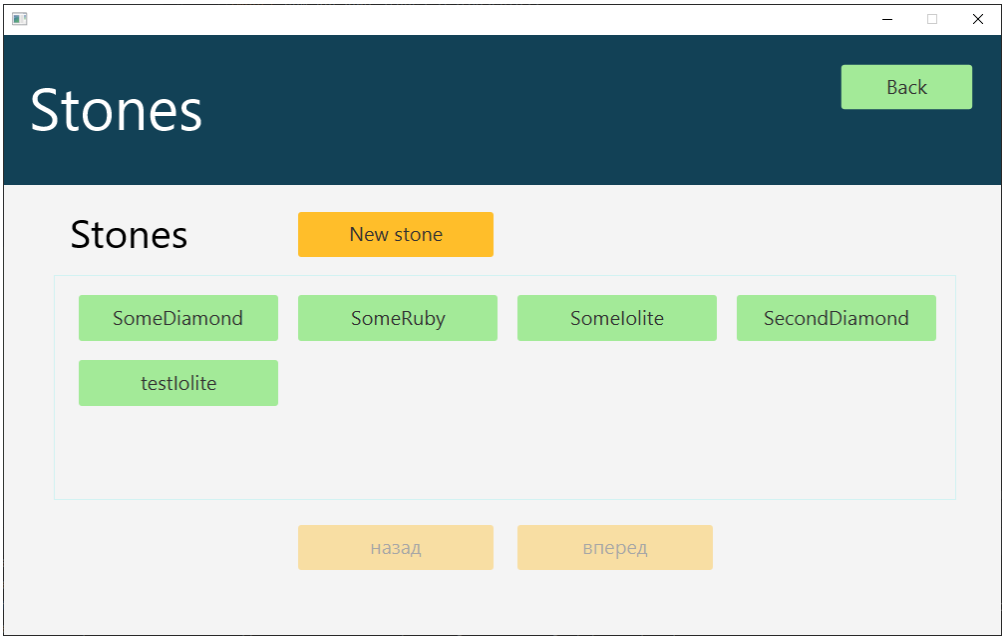

Рис. 4.2.1 – Пункт меню камені під час тестування

Обрав пункт створення каменю та заповнив поля, залишивши одне з них пустим. При спробі зберегти камінь отримую діалогове вікно з заміткою, що всі поля повинні бути заповненими.

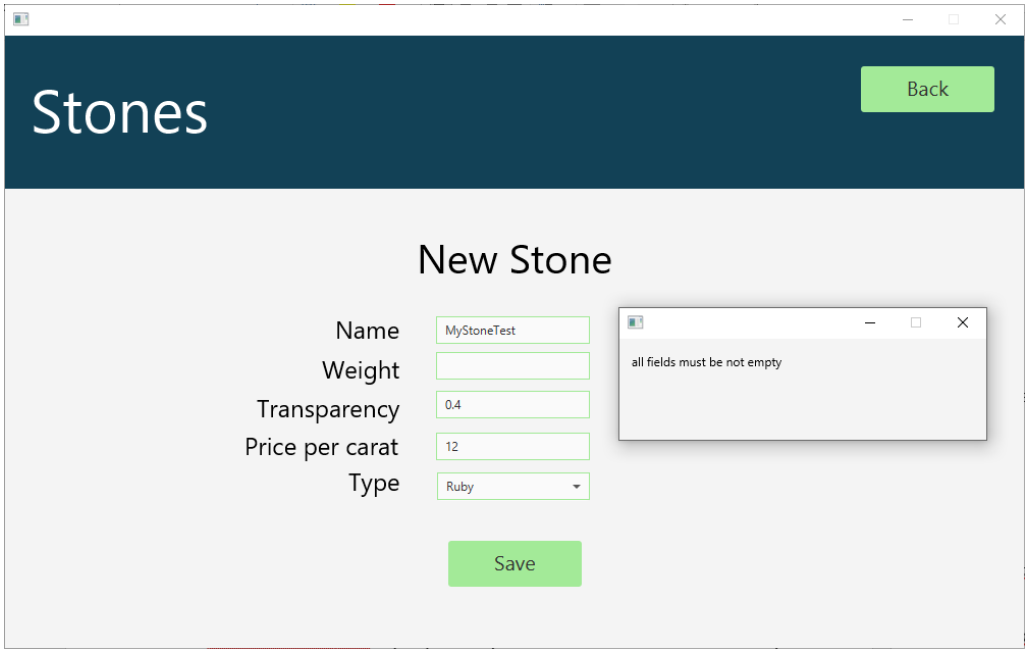

Рис. 4.2.2 – Спроба зберегти камінь з пропущеними значеннями полів

Додав значення ваги. Оскільки значення прозорості вимірюється у відсотках, пробую зберегти камінь зі значенням, що не задовільняє умову. Отримую діалогове вікно з зауваженням, що значення прозорості має бути

меншим за 1. Як щодо від’ємного значення? Поле налаштоване так, що ввести від’ємне значення неможливо.

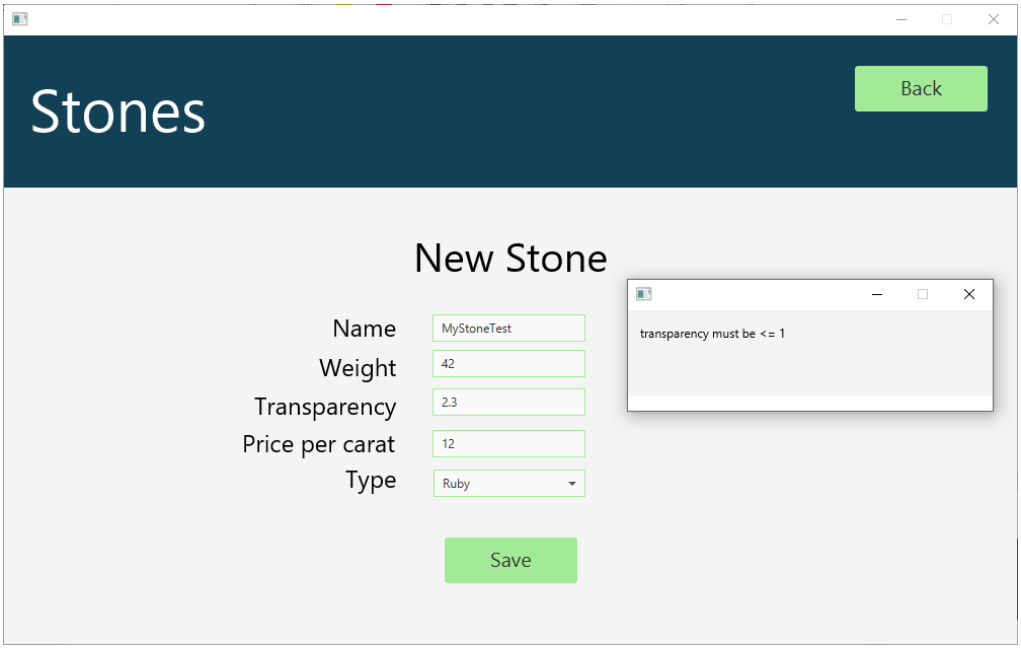

Рис. 4.2.3 – Спроба зберегти камінь з значенням прозорості < 1

Провів схожі експерименти з меню редагування каменю. Додав ще кілька каменів, щоб їхня кількість була більшою за 12 та можна було протестувати можливість пагінації.

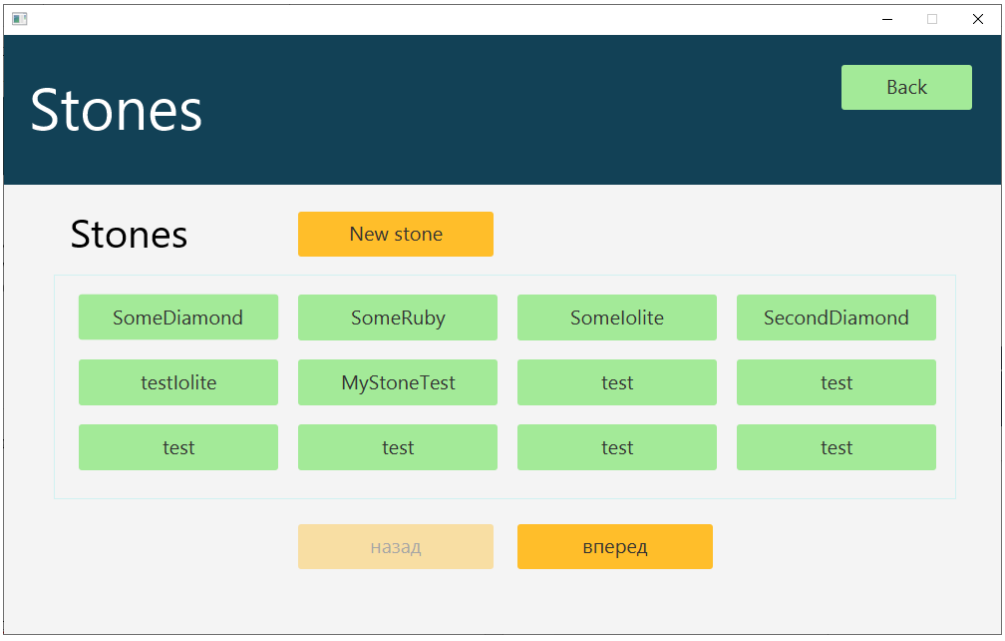

Рис. 4.2.4 – Меню каменів з активною пагінацією сторінка 1

Як тільки кількість каменів перевалила за 12 розблокувалася кнопка переходу до наступної сторінки, що дозволяє зберігати безмежну кількість каменів.

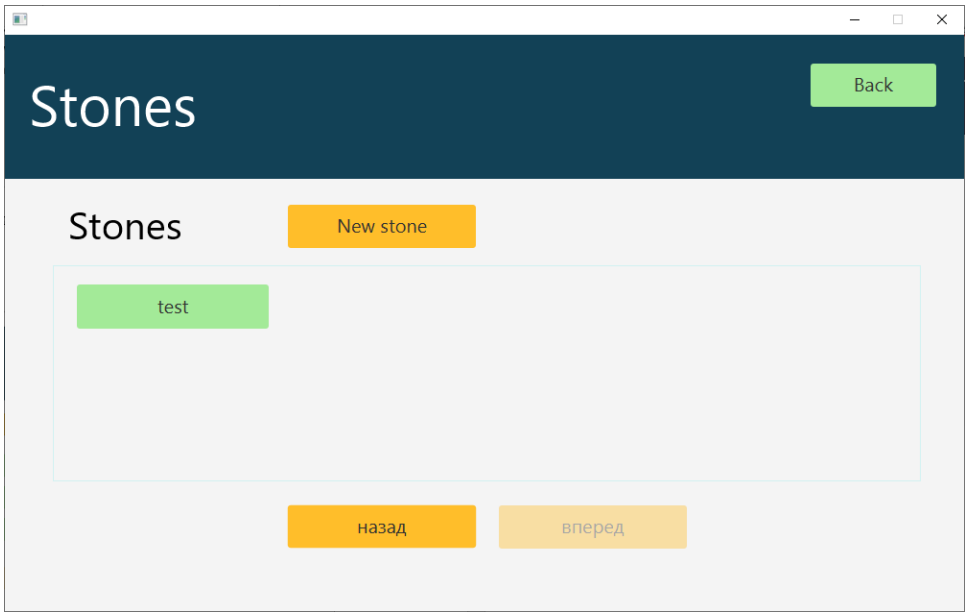

Рис. 4.2.5 – Меню каменів з активною пагінацією сторінка 2

Повернувся до головного меню та перейшов до меню намист, воно використовує ту ж сцену з тією ж логікою пагінації, та кнопка, яка у випадку каміння відповідала за створення нового каменю тепер відкриє вікно створення Намиста.

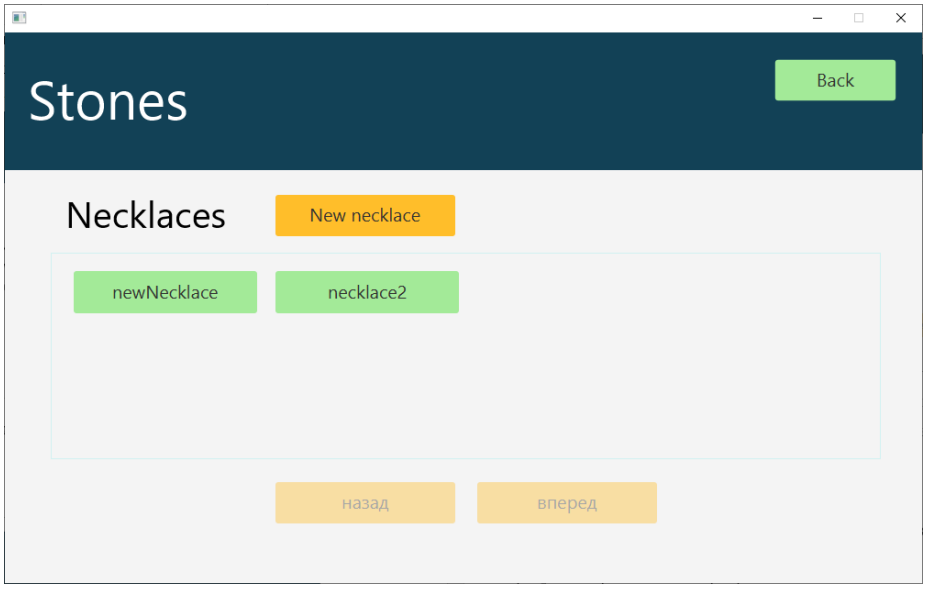

Рис. 4.2.6 – Меню намист

Відкриваю вікно створення намиста. Також пробую ввести невірні дані та зберегти. Знову ж отримую діалогове вікно з помилкою.

Рис. 4.2.7 – Спроба створити намисто без назви

Повертаючись до намист, обрав заздалегідь підготоване намисто2. Тут відображаються всі камені, що належать намисту у порядку додавання.

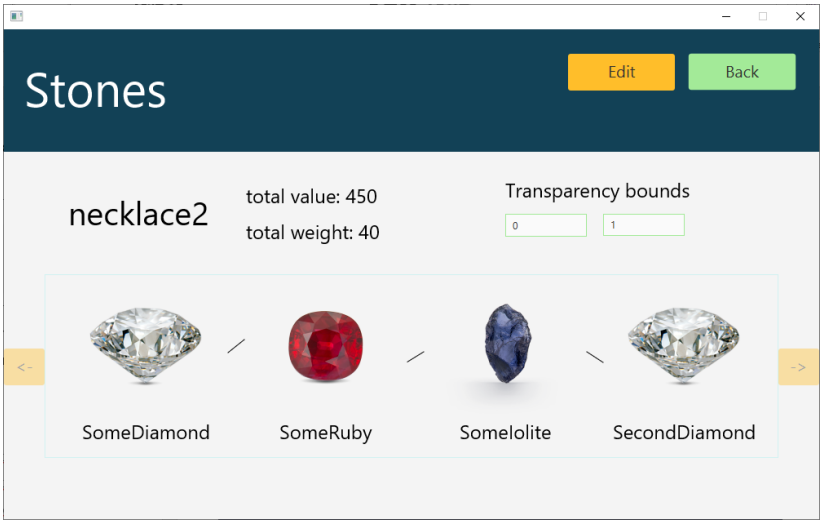

Рис. 4.2.8 – Намисто до сортування

Додам ще один камінь щоб мати змогу перейти до наступної сторінки. Після цього одразу змінилася ціна та вага намиста.

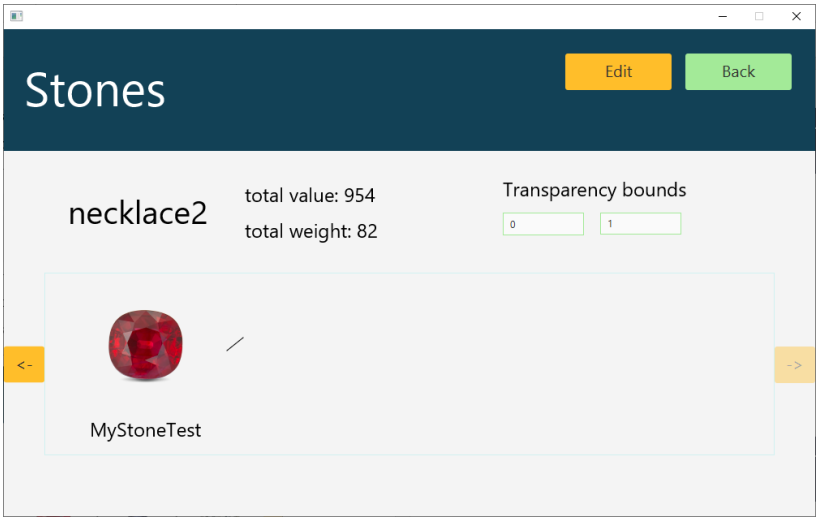

Рис. 4.2.9 – Намисто сторінка каменів 2

Перевіряю пошук за прозорістю. Він працює автоматично, при кожній зміні значень полів для позначення граничних значень прозорості.

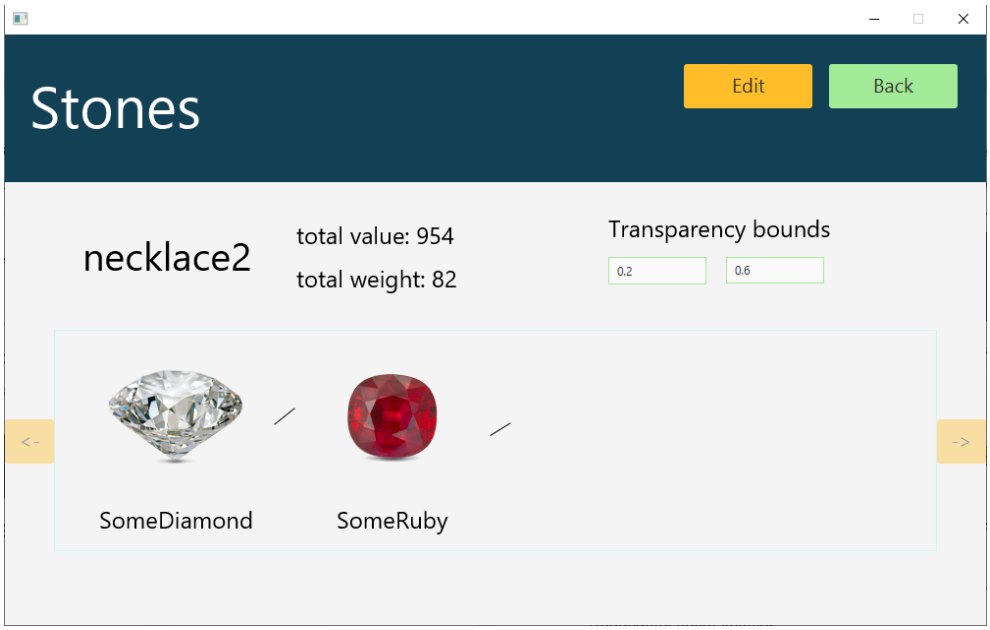

Рис. 4.2.10 – Пошук за прозорістю

Переходжу у меню редагування намиста. Тут є можливість додати до намиста камінь, що не належить жодному намисту, сортувати каміння в намисті, та повністю його видалити.

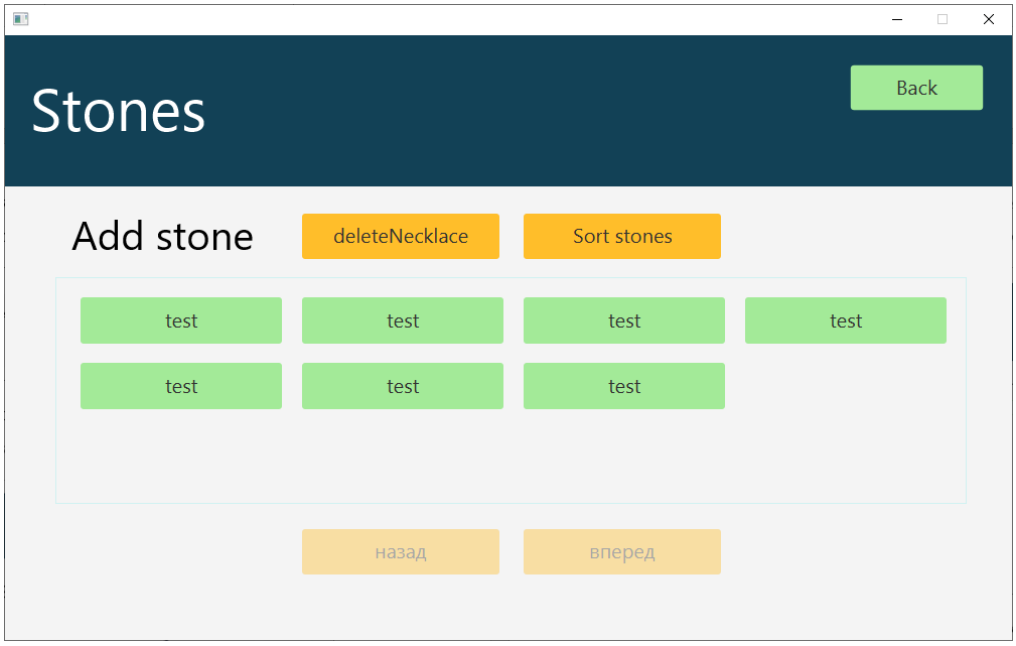

Рис. 4.2.11 – Меню редагування намиста

При обранні пункту сортування каміння отримую діалогове вікно, у якому зазначено за яким принципом їх було відсортовано, а також переходжу до меню намиста, де можу переглянути результат сортування.

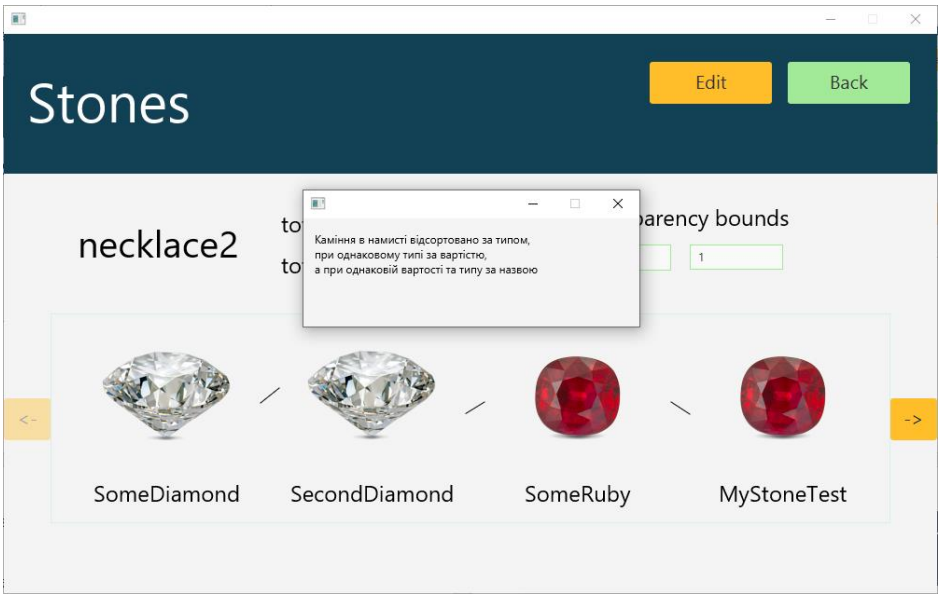

Рис. 4.2.12 – Намисто після сортування

В програмі закладено, що алмаз найрідкісніший тип, рубін дещо більш розповсюджений, а іоліт найрозповсюдженіший серед них. Щоб видалити камінь з намиста, обираю який саме камінь потрібно видалити та обираю пункт видалити з намиста. Переходжу у меню редагування та обираю видалити намисто.

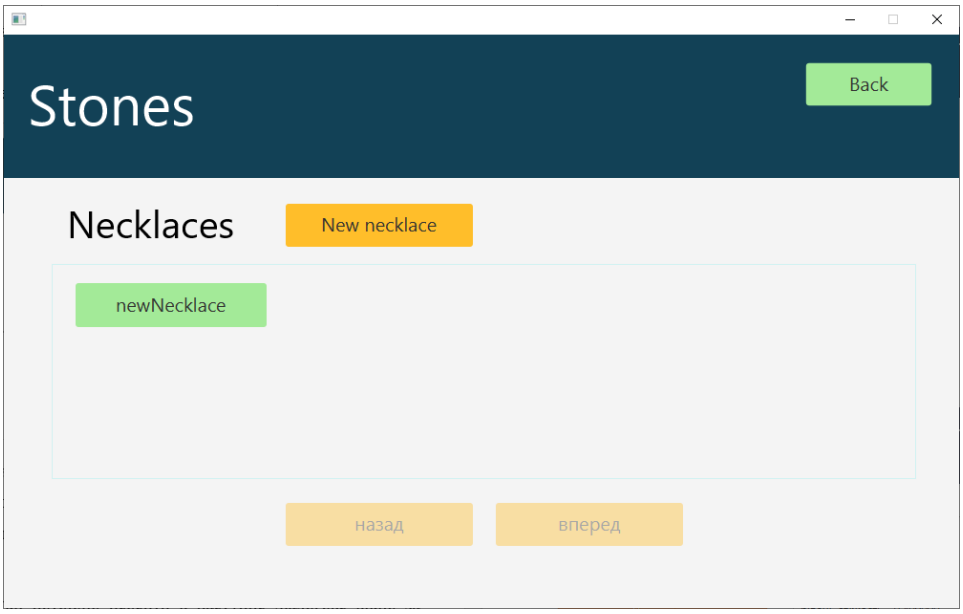

Рис. 4.2.13 – Меню намист після видалення

4.3. Аналіз результатів.

Використовую вбудований в intelij idea засіб перевірки покриття коду тестами.

Цей показник складає 88% для класів програми, 79% для методів, 83% для рядків коду та 80% для сценаріїв.

Element	Class, %	Method, %	Line, %	Branch, %
com.example.diploma	88% (24/27)	79% (105/132)	83% (299/356)	80% (64/80)
comparators	100% (1/1)	100% (1/1)	100% (5/5)	100% (4/4)
controllers	100% (18/18)	84% (77/91)	88% (260/294)	87% (42/48)
messages	0% (0/1)	100% (0/0)	100% (0/0)	100% (0/0)
models	83% (5/6)	75% (27/36)	79% (34/43)	64% (18/28)
repositories	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
StonesApplicationNative	0% (0/1)	0% (0/4)	0% (0/14)	100% (0/0)

Рис. 4.3.1 – Покриття коду тестами.

Добре протестованою програмою вважається та, 80% рядків якої покрито тестами, очевидно це працює не у всіх випадках, також варто розглянути й інші показники і окремі випадки.

Також гарною практикою є проведення BDD (Behavior Driven Development). Цей метод тестування полягає у створенні та реалізації тестових сценаріїв, схожих з діями користувача. Ця система здебільшого тестується саме BDD тестами.

Відповідно до цієї інформації роблю висновок, що програму протестовано на хорошому рівні.

РОЗДІЛ 5
ЕКОНОМІЧНА ЧАСТИНА

5.1. ВИБІР ТА ОБҐРУНТУВАННЯ БАЗОВОГО ПРОГРАМНОГО ПРОДУКТУ

У якості базового варіанту (надалі «бази») повинен бути обраний найбільш досконалий програмний продукт даного призначення, що перевершує за своїм рівнем та економічністю інші відомі зразки.

Не можна використовувати у якості бази для розрахунків гіпотетичні (уявні) зразки. Використання застарілих і недосконалих програмних продуктів призводить до перекручення, необґрунтованого завищення ефективності створюваного нового програмного продукту, хоч саме такі зазвичай і використовуються у цій сфері. Для обґрунтованого вибору бази необхідно, як мінімум, запропонувати для аналізу три аналогічних програмних продукти.

Визначення бази здійснюється методом порівняння технічних та економічних показників аналогічних продуктів з використанням розрахункових технічних характеристик. Розрахунок відносних характеристик та економічних показників необхідно узагальнити у таблиці 1.1.

Таблця 5.1

Технічні та економічні показники аналогічних програмних продуктів

№ з/п	Найменування аналогічних програмних продуктів	Технічні показники				Економічні показники				Відносні показники			
		1	2	3	4	5	6	7	8	Q1	Q2	Q3	Q4
1	Stonny	130	4	3	1	-	1	4560	3	1.3	0.5	1	0.7
2	Dilovod	130	3	3	1	-	1	1200	5	1.3	0.66	1	0.92
3	Storager	100	2	3	1	-	1	6238.5	2				

У якості технічних та економічних показників програмного продукту, що дозволяють порівнювати його з аналогічними були прийняті дані, наведені у таблиці 5.2.

Для оцінки нашої дипломної роботи в якості технічних показників були взяті наступні:

Таблиця 5.2

Технічні характеристики та економічні показники програмного продукту

№ з/п	Характеристика	Одиниця виміру
1	**2**	**3**
1	Середня швидкість завантаження веб сторінки	Сек
2	Складність програмного продукту:	
	- складність інтерфейсу користувача	кількість розділів *та/або* кількість керуючих елементів на сторінці
	- виконувані функції	кількість
	- обсяг сторінок	кількість
	- обсяг виконуваного коду	Мбайт
3	Доступність використання	
	- по інтерфейсу	кількість балів: 2 – палітра кольорів; 3 – шрифти; 4 – адаптований інтерфейс; 5 – загальний інтерфейс;
	- по простоті користування	рівень запуску: - низький – без навігаційної панелі; - середній – неправильні назви посилань; - високий – чіткий перехід та навігаційна панель;
	- по мультимедійному забезпеченню	рівень запуску: - низький – мультимедіа відсутня; - середній – звукове супроводження; - високий – інтерактивне мовне спілкування;
4	Супровідна документація	є / немає
5	Засоби навчання	кількість розділів
6	Довідкова система	кількість розділів
7	Ціна	грн.
8	Термін служби веб сайту	Років

Відносні безрозмірні показники встановлювались диференціальним методом як відношення одиночного технічного або економічного показника P_i до

відповідного показника умовно-базової конструкції $P_{iб}$. Якщо збільшення показника сприяє поліпшенню якості продукту, то відносний показник розраховували за формулою 5.1:

$$q_i = \frac{P_i}{P_{iб}} \tag{5.1}$$

Якщо зменшення показника характеризує покращення якості продукту, то відносний показник розраховують за формулою 5.2:

$$q_i = \frac{P_{iб}}{P_i} \tag{5.2}$$

Тобто, для продукту (1) і продукту (2) отримуємо:

$$q_1^{(1)} = \frac{130}{100} = 1{,}30; \quad q_2^{(1)} = \frac{2}{4} = 0{,}50; \quad q_3^{(1)} = \frac{3}{3} = 1{,}00;$$

$$q_1^{(2)} = \frac{130}{100} = 1{,}30; \quad q_2^{(2)} = \frac{2}{3} = 0{,}66; \quad q_3^{(2)} = \frac{3}{3} = 1{,}00;$$

Різновидність бальної оцінки – використання узагальнюючого показника технічного рівня нового продукту $q_{m.y}$, який визначається як добуток окремих відносних показників q_i:

$$q_{m,y} = \prod_{i=1}^{i=n} q_i \tag{5.3}$$

Тобто

$$q_{m,y}^{(1)} = 0{,}71 * 0{,}80 * 1 = 0{,}57$$

5.2 Розрахунок собівартості нового програмного продукту

Собівартість програмного продукту – це виражені у грошовій формі витрати на його розробку і, при необхідності, реалізацію.

Форма кошторису витрат на розробку програмного продукту наведена у таблиці 5.3.

Таблиця 5.3

Кошторис витрат на розробку програмного продукту

№ з/п	Найменування витрат за економічними елементами	Розмір, грн.	Підстава
1	2	3	4
1	Витрати на оплату праці, у тому числі:	1 506,45	
1.1	Заробітна плата керівника дипломної роботи та консультантів	8989,2 8989,2 9583,2	Розрахунок за формулами
1.2	Стипендія	0	Фактична
2	Відрахування на соціальні заходи	331,4	22,0% від п.1.1
3	Матеріальні витрати	365,00	Розрахунок за текстом методичних рекомендацій
4	Амортизація обладнання	612,8	Розрахунок за текстом методичних рекомендацій
5	Витрати на електроенергію	256,0	Розрахунок за текстом методичних рекомендацій
6	Витрати на роботи, які виконують сторонні організації	140,00	Розрахунок за квитанціями
7	Витрати на машинний час	6998,25	Розрахунок за текстом методичних рекомендацій
8	Накладні витрати	753,22	Розрахунок за формулами
	Разом собівартість програмного продукту	11716,34	

5.2.1 Визначення розміру витрат на оплату праці

Середньо-годинна ставка зарплати ($Сг_i$) для кожного з виконавців буде визначатися за формулою 5.4:

$$C_{Гi} = ЗП_i/F_p \quad (5.4),$$

де $F_р$ – місячний фонд робочого часу (72 години).

$$Сг_{і\text{-кер}} = 8989,2 / 72 = 124,85 \text{ (грн.)};$$
$$Сг_{і\text{-конс е.ч.}} = 8989,2 / 72 = 124,85 \text{ (грн.)};$$
$$Сг_{іконс о.п.} = 9583,2 / 72 = 133,1 \text{ (грн.)}.$$

Для розрахунку витрат на оплату праці виконавців даного дипломної роботи (табл. 5.4) визначалася трудомісткість роботи кожного з працівників (Ті) (в людино-годинах), виходячи з діаграми завантаження виконавців.

Діаграма завантаження виконавців ДП (Ті, люд./год.):

1. Керівник ДП – керівництво ДП – 10 год.
2. Консультант з економіки – консультації – 1 год.
3. Консультант з охорони праці – консультації – 1 год.
4. Дипломник – 30 днів * 8 год. = 240 год.

Таблиця. 5.4

Розрахунок витрат на оплату праці виконавців дипломної роботи

№ з/п	Посади виконавців	$С_{гі}$ грн./год.	$Т_і$, люд./год.	Витрати на оплату праці (Воп), грн.
1	2	3	4	5
1	Керівник ДП	124,85	10	1248,5
2	Консультант з економіки	124,85	1	124,85
3	Консультант з охорони праці	133,1	1	133,1
4	Дипломник	-	240	0
	Разом	-	-	1 506,45

Отже, витрати на оплату праці становлять 1 506,45 грн.

5.2.2 Відрахування на соціальні заходи

Розрахунок відрахувань здійснюється за наступною формулою:

$$B_{єсв}=0,22*B_{оп}, \quad (5.5)$$

$B_{оп}$ – загальна сума оплати праці без стипендії (нарахування єдиного соціального внеску на стипендію не проводиться).

$$B_{єсв}=0,22 * 1\,506,45 = 331,42 \text{ (грн.)}$$

Отже, відрахування на соціальні заходи становлять 331,4 грн.

5.2.3 Визначення розміру матеріальних витрат

Визначення витрат на матеріали та окремі комплектуючі вироби, визначається з урахуванням вирішення поставленого конкретного завдання – розробки програмного продукту.

Таблиця 5.5

Розрахунок витрат матеріалу

№ з/п	Найменування (вид) матеріалу	Кількість, (один.)	Ціна, (грн.)	Сума (грн.)
1	2	3	4	5
1	Папір, формат А4	1 пачка (500 арк.)	250,00	250,00
2	Фарба для принтера	1 банка	100,00	100,00
3	Диск (CD-R)	1 диск	15,00	15,00
	Разом			365,00

Отже, розмір матеріальних витрат становить 365,00 грн.

5.2.4 Визначення розміру амортизаційних відрахувань

Визначення розміру амортизаційних відрахувань (А) визначається за формулою 5.6:

$$A = \frac{K_b * N_a}{\text{Ткор} * 100} * T_{рм}, \quad (5.6),$$

де $К_б$ – балансова вартість однієї ПЕОМ з периферією,

$N_а$ – річна норма амортизаційних відрахувань,

$Т_{рм}$ – тривалість розробки програмного продукту у місяцях,

$Т_{кор}$ – термін корисного використання.

Річні норми амортизаційних відрахувань для ПЕОМ приймаються у розмірі 25%, а для ПЗ – 60%.

$$T_{рм} = \frac{t_ч}{T_{міс}} \quad (5.7)$$

$$T_{рм} = \frac{30}{31} = 0{,}96 \text{ (міс.)}$$

Вартість ПК становить 35000,00 грн., принтера – 6090,00 грн. Отже:

$$А_{(ПЕОМ)} = \frac{35000{,}00*25}{60*100} * 0{,}96 = 145{,}83 \text{ (грн.)}$$

$$А_{(принтер)} = \frac{6090{,}00*25}{60*100} * 0{,}96 = 25{,}37 \text{ (грн.)}$$

Вартість IDE Intelij Idea становить 5000 грн., Windows становить 2500,00 грн., а Microsoft Office – 1700,00 грн.

$$К_{б(ПЗ)} = 5000{,}00 + 2500{,}00 + 1700{,}00 = 9200{,}00 \text{ (грн.)}$$

Розмір амортизаційних відрахувань для ПЗ:

$$А_{(ПЗ)} = \frac{9200{,}00 * 60}{12*100} * 0{,}96 = 441{,}60 \text{ (грн.)}$$

Загальний розмір амортизаційних відрахувань:

$$А_{(заг)} = 145{,}83 + 25{,}37 + 441{,}60 = 612{,}80 \text{ (грн.)}$$

Отже, розмір амортизаційних відрахувань становить 612,8 грн.

5.2.5 Визначення витрат на електроенергію

Розмір витрат на електроенергію включає:

- витрати на силову електроенергію;
- витрати на електроенергію, яка витрачається на освітлення, та визначається по формулах 5.8, 5.9.

Витрати на силову електроенергію (грн.) визначаються за формулою:

$$З_{с.е} = Т^{год}_{рп} * Ц_е * Р_{ОЕМ,} \quad (5.8),$$

де $Ц_е$ – вартість 1кВт/год. – 4,32 грн.,

$Р_{ОЕМ}$ – сумарна потужність для ПЕОМ з периферією у кіловат-годинах: для ПК – 0,38 кВт/год. та для принтера – 0,12 кВт/год.

$Т^{год}_{рп}$ – тривалість розробки програмного продукту у годинах (час використання ПК – 155 год. та час використання принтера – 3 год.).

$$З_{с.е(ПК)} = 155 * 4,32 * 0,38 = 254,45 \text{ (грн.)}$$
$$З_{с.е(принтер)} = 3 * 4,32 * 0,12 = 1,55 \text{ (грн.)}$$

Загальні витрати на силову енергію:

$$З_{с.е} = 254,45 + 1,55 = 256,0 \text{ (грн.)}$$

Витрати на електроенергію для освітлення визначаються за формулою 5.9:

$$З_{ос} = Т^{год}_{рп} * Ц_е * Р_{осв} \quad (5.9),$$

де $Р_{осв}$ – сумарна потужність у кіловат-годинах, яка йде на освітлення.

При виконанні дипломної роботи штучне освітлення не використовувалося.

Загальні витрати на електроенергію складають:

$$З_{загальне} = З_{с.е.} + З_{ос}, \quad (5.10)$$

$$З_{загальне} = 256,0 + 0 = 256,0 \text{ грн.}$$

Отже, загальні витрати на електроенергію становлять 256,0 грн.

5.2.6. Розрахунок витрат на роботи сторонніх організацій

До цієї статті належать витрати ($В_{со}$) на виконання окремих робіт для даного дипломної роботи в силу відсутності потрібного обладнання або відповідних спеціалістів і тому виконуються на договірній основі з іншими організаціями.

$В_{со}$ розраховується за формулою 5.11:

$$В_{со} = N*Ц \quad (5.11),$$

де N – кількість робіт (послуг), згідно квитанції,

Ц – ціна 1 роботи (послуги), згідно квитанції.

При розробці даної дипломної роботи було використано наступні послуги сторонніх організацій, як:

1. Вартість брошурування дипломної роботи (300,00 грн.);
2. Вартість роздруку широкоформатних додатків А1 (2 додатки по 30,00 грн.).

Витрати на роботи, які виконують сторонні організації:

$$В_{со}=1*300,00 + 2*30,00= 360,00 \text{ (грн.)}$$

Отже, витрати на роботу, які виконують сторонні організації становлять 360,00 грн.

5.2.7 Розрахунок витрат на машинний час

Розрахунок витрат на машинний час здійснюється за формулою:

$$C_{маш.ч} = Ц_{маш.ч} * T_{маш.ч} \quad (5.12),$$

де $Ц_{маш.ч}$ – собівартість однієї години машинного часу

$T_{маш.ч}$ – машинний час, використаний для проведення робіт.

Для дипломної роботи приймаємо $Ц_{маш.ч} = 45,15$ грн.

Необхідна кількість машинного часу для реалізації роботи з розробки програми розраховується за формулою:

$$T_{маш.ч.} = T_i * t_З * T_{ср.маш.}, \quad (5.13),$$

де T_i – трудомісткість робіт, люд.дн,

$t_З$ – тривалість робочої зміни (при п'ятиденному робочому тижні $t_З = 5,17$ год.),

$T_{ср.маш}$ – середній коефіцієнт використання машинного часу ($T_{ср.маш} = 1$).

Машинний час, використаний для проведення робіт:

$$T_{маш.ч.} = 30*5,17*1 = 155 \text{ (год.)}$$

Витрати на машинний час:

$$C_{маш.ч} = 45,15 * 155 = 6998,25 \text{ (грн.)}$$

Отже, витрати на машинний час складають 6998,25 грн.

5.2.8 Розрахунок накладних витрат

До складу накладних витрат ($В_н$) відносяться:

- витрати, пов'язані з управлінням організацією, де виконується дипломна робота;
- витрати на науково-технічну інформацію;
- витрати на забезпечення нормальних умов праці і техніки безпеки;
- витрати на інші загальногосподарські потреби, тощо.

Накладні витрати розраховуються у відсотках до витрат на оплату праці ($В_{оп}$) і визначаються за формулою 5.14:

$$B_н = \frac{\alpha}{100} * B_{оп}, \qquad (5.14),$$

де α – середньостатистичний відсоток накладних витрат в організації (50%).

$$В_н = \frac{50}{100} * 1\,506{,}45 = 753{,}22 \text{ (грн.)}$$

Отже, накладні витрати ($В_н$) становлять 753,22 грн

5.3 Визначення ціни програмного продукту

Ціна програмного продукту $Ц_{пп}$ визначається за формулою 5.15:

$$Ц_{пп} = С_{пп} + П \qquad (5.15),$$

де $С_{пп}$ – собівартість програмного продукту (грн.);

П – прибуток, що планується (грн.) (25% від собівартості):

$$Ц_{пп} = 11716{,}34 + 2\,929{,}08 = 14\,645{,}42 \text{ (грн.)}$$

Висновок

В економічній частині дипломного проекту проведено розрахунок витрат на розробку програмного продукту “Програма для обліку дорогоцінного каміння з можливістю створення виробів”. Сума цих витрат складає 14 645,42 грн.

РОЗДІЛ 6 ОХОРОНА ПРАЦІ

Вступ

Охорона праці – це система правових, соціально-економічних, організаційно технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Створення безпечних і здорових умов праці сприяє підвищенню її продуктивності та зниженню собівартості продукції. Підвищення продуктивності відбувається за рахунок зниження стомлюваності працюючих протягом робочого часу, його раціонального використання Прогресивне суспільство дає можливість використання. техніки на користь робітників для підвищення рівня життя, для оздоровлення і поліпшення умов праці. Для аналізу умов праці вибрано підприємство ТОВ «ЛЛРЗ».

6.1. Аналіз умов праці

6.1.1. Організація робочого місця

Підприємство розташоване в одноповерховій будівлі і займає 1 кімнату. Приміщення, в якому розроблявся проект, має розмір 7 х 5х 3 м. Площа приміщення становить 35 м^2, об'єм 105 м^3. Штат працівників в даному приміщенні – 3 особи. Таким чином на одну людину припадає 12 м^2 площі приміщення та 35 м^3 його об'єму. На одного працюючого встановлено об'єм виробничого приміщення не менше 19,5 м^3 та площу приміщення не менше 6 м^2 на одного працюючого. Отже, умови для даного приміщення виконуються.

В приміщенні встановлено 3 комп'ютера, принтер та ксерокс.

Відповідно до наказу МСО України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» від 14.02.2018 № 207 встановлено такі вимоги до безпеки робочих місць працівників з екранними пристроями:

– робочі місця працівників з екранними пристроями мають бути

достатнього розміру, щоб працівники мали простір для зміни робочого положення та рухів.

- все випромінювання від екранних пристроїв має бути зведене до гранично допустимого рівня з погляду безпеки та охорони здоров'я працівників.
- організація робочого місця має відповідати ергономічним, антропологічним та психофізіологічним вимогам.
- освітлення робочого місця повинно створювати відповідний контраст між екраном і навколишнім середовищем.
- мікроклімат виробничих приміщень має відповідати вимогам Санітарних норм мікроклімату.
- робочий стіл та крісло повинні бути зручними, а робоча поверхня - достатнього розміру.
- робоче крісло має дозволяти працівнику легко рухатися та займати зручне положення.

Також у цьому наказі йдеться про мінімальні вимоги безпеки під час роботи з екранними пристроями:

- Щодня перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень.
- Після закінчення роботи екранні пристрої слід відключати від електричної мережі.
- У разі виникнення аварійної ситуації необхідно негайно відключити екранний пристрій від електричної мережі.
- Не допускається:
    - виконувати технічне обслуговування, ремонт і налагодження екранних пристроїв безпосередньо на робочому місці працівника під час роботи з екранними пристроями;
    - відключати захисні пристрої, самочинно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;

- працювати з екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані та інші несправності.

– Під час виконання робіт операторського типу, пов'язаних з нервово-емоційним напруженням, у приміщеннях під час роботи з екранними пристроями, на пультах і постах керування технологічними процесами та в інших приміщеннях мають дотримуватися оптимальні умови мікроклімату відповідно до вимог ДСН 3.3.6.042-99.

6.1.2. Мікроклімат виробничих приміщень

Метеорологічні умови середовища та характер виконуваної роботи створюють мікроклімат, який впливає на процес теплового обміну людського організму із зовнішнім середовищем. Мікроклімат визначається температурою, вологістю та швидкістю руху повітря, що діють на людину. Згідно ДСН 3.3.6.042- 99 «Санітарні норми мікроклімату виробничих приміщень» мікроклімат робочої зони нормується залежно від періоду року та категорії робіт за енерговитратами.

За енерговитратами роботи, що виконуються в процесі виконання дипломного проекту, належать до легких (витрати менше 150 ккал/год).

Таблиця 6.1

Параметри мікроклімату

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря приміщення	18-22°С
	Відносна вологість	50-60%
	Швидкість руху повітря	До 0,3 м/с
Теплий	Температура повітря приміщення	22-25°С
	Відносна вологість	40-60%
	Швидкість руху повітря	0,1-0,2 м/с

6.1.3. Шкідливі речовини в повітрі робочої зони

При роботі лазерного принтера виділяється озон, оксиди азоту (NO_2,N_2O), ацетон, паперовий пил, ультрафіолетове і інфрачервоне (теплове) випромінювання. При нагріванні паперу активно випаровується формальдегіди і водяна пара.

6.1.4. Освітлення

Приміщення має природне освітлення з КПО 1.5, який нормується ДБН В.2.5-28-2006 («Природне та штучне освітлення»).

Для роботи зранку та ввечері в приміщенні передбачене також штучне освітлення.

Оскільки в приміщенні проводяться, в основному, роботи з використанням обчислювальної техніки, то згідно ДБН В.2.5-28-2006 мінімальна освітленість повинна становити 300 лк.

Розрахунок штучного освітлення проведемо користуючись методом світлового потоку.

Даний метод дозволяє визначити світловий потік ламп, необхідний для досягнення заданої освітленості із урахуванням світла відбитого від стін, стелі та робочої поверхні.

Світловий потік розраховується за формулою:

$$F = \frac{Eн\ S\ Kз\ Z}{Nn\eta}, \qquad (6.1),$$

де $E_н$ – нормована освітленість, вибирається з таблиці 2.4 (400 лк);

$K_з$ – коефіцієнт запасу, вибирається з таблиці 2.5 (1,3);

Z – коефіцієнт мінімальної освітленості, що дорівнює відношенню середньої освітленості до мінімальної (приймається рівним 1,1 ... 2);

– коефіцієнт використання світлового потоку, вибирається з таблиці 2.8. Він залежить від показника приміщення, розподілу сили світильника, коефіцієнтів відбиття потоку $\rho_п$ – від стін; $\rho_с$ – стелі; $\rho_{РП}$ - робочої поверхні. Для знаходження розрахуємо показник приміщення і:

I=B * A/H*(B+A) (6.2),

де, A і B - відповідно довжина та ширина приміщення в метрах (7м, 5м);

H - висота підвісу світильника над робочою поверхнею (3 м).

Для даного приміщення i = 5 * 7 /(3 *(5+7)) = 1,26.

Прийнявши ρп=50%, ρс = 30%, ρРП= 10%, що для i = 1,26 та світильника типу EGLO 12720 η=46%.

Знаходимо F_Σ:

F_Σ= (348*35*1,26*1,26/0.46) = 42036,88 (лм).

Необхідна кількість ламп розраховується за формулою:

$$N = \frac{F_\Sigma}{F_л} \quad (6.3)$$

де Fл - світловий потік однієї лампи (лм).

Для освітлення вибираємо лампу, у якої Fл= 2740 лм при напрузі живлення 220 В. Тоді

n = 42036,88 / 2740 ≈ 15 (ламп).

У світильнику EGLO 12720 використано три лампи ELGO 96431 і загальна кількість, світильників рівна 5 , що відповідає дійсності.

Перевіримо правильність розрахунку. Для цього знайдемо освітлення, що створюється вибраними світильниками за наступною формулою:

$$E = \frac{F_н \cdot \eta \cdot n \cdot N}{S \cdot K \cdot Z} \text{ (лк)} \quad (6.4)$$

E= (2740*0,46*3*5)/(35*1,26*1,26)= 340,24 (лк)

Отже, розрахована система освітлення забезпечує рівень освітленості вищий за мінімально допустимий: 340лк > 300лк.

Схема розташування світильників приведена на рис. 6.1.

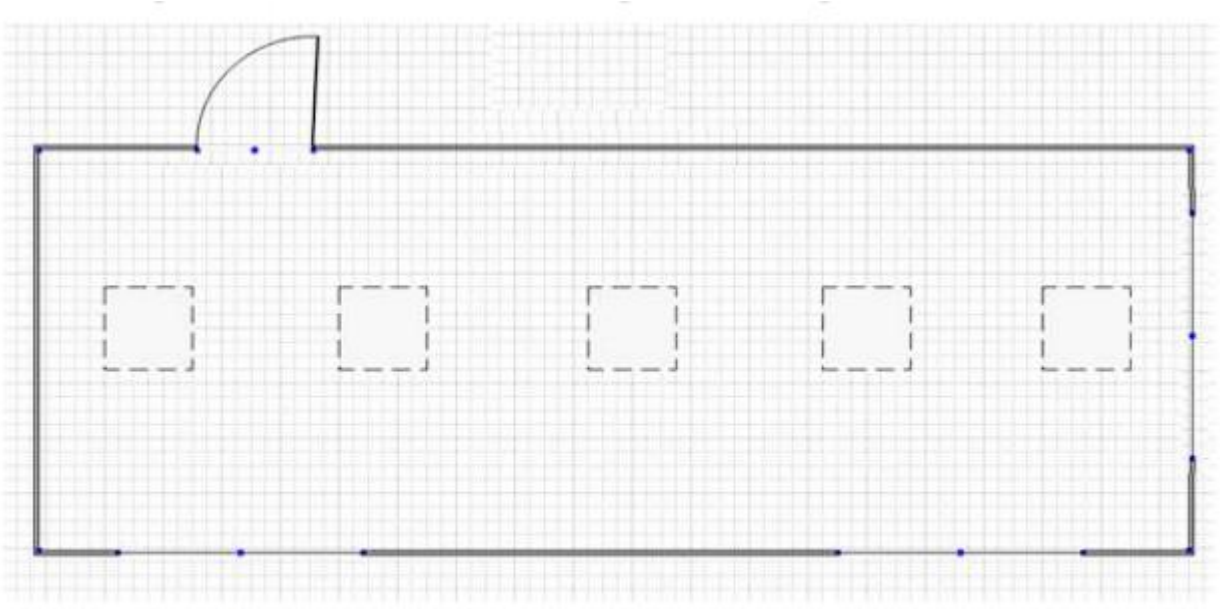

Рис. 6.2 - Схема приміщення та розміщення світильників

6.1.5. Шум, вібрація, ультразвук

Джерелами шуму у приміщенні являються вентилятори системних блоків комп'ютерів. Остання перевірка рівня шуму (квітень 2019 р.) при паспортизації підприємства показала, що еквівалентні рівні шуму на робочих місцях не перевищують 39 дБА, що відповідає загальним вимогам безпеки, встановленим ДСН 3.3.6.037-99 "Санітарні норми виробничого шуму, ультразвуку та інфразвуку" . Згідно вказаних вимог еквівалентні рівні шуму на робочих місцях, де працюють програмісти та оператори ЕОМ, не повинні перевищувати 50 дБА

6.1.6. Виробничі випромінювання

Джерелами випромінювання електромагнітної енергії радіочастотного діапазону є монітори. Електромагнітні поля можуть негативно впливати на організм людини. Для захисту персоналу від їх шкідливої дії використовуються монітори із зниженою випромінювальною здатністю, віддалення робочого місця

на безпечну відстань, зменшення часу перебування у небезпечній зоні, застосування засобів індивідуального захисту, а також дотримання регламентованого режиму праці і відпочинку.

6.1.7. Небезпека ураження електричним струмом

Приміщення, в якому розроблявся дипломний проект, відноситься до сухих приміщень, із нормальною температурою й вологістю повітря, а також має ізольовану підлогу, тому за небезпекою ураження електричним струмом вона відноситься до першого класу, тобто приміщення без підвищеної небезпеки. В приміщенні використовується однофазна мережа частотою 50 Гц, напругою 220 В з використанням третього захисного провідника, з'єднаного з землею.

Покриття поверхні стола матове з коефіцієнтом відбиття 10%, легко чиститься, кути і передня панель дошки стола заокруглені. Сидіння нахиляється по відношенню до горизонталі вперед на 20° і назад на 140°, розмір його 50 х 45 см. Висота спинки крісла складає 55 см від поверхні сидіння.Згідно з ДСТУ 7237:2011 приміщення відноситься до класу приміщення без підвищеної небезпеки. Приміщення без підвищеної небезпеки характеризуються відсутністю умов, що створюють особливу або підвищену небезпеку. В приміщенні використовується однофазна мережа частотою 50 Гц, напругою 220 В. Підвищення електробезпеки досягається застосуванням систем захисного заземлення, занулення, захисного відключення й інших засобів і методів захисту, у тому числі знаків безпеки і попереджувальних плакатів і написів.

6.1.8. Ергономіка, технічна естетика

Основне робоче місце – це комп'ютерний стіл, на якому розташований персональний комп'ютер. При організації робочого місця враховані антропометричні дані працівників. Робочий стіл має стабільну конструкцію: площина стола складає 140 х 80 см, висота столу 80 см, висота від горизонтальної лінії зору до робочої поверхні стола складає 45 см. Висота сидіння регулюється по висоті в межах 42-55 см.

Рис. 6.1 - Схематичний малюнок ергономіки робочого місця

Використання чинників виробничого естетичного впливу на працівників у значній мірі може привезти до підвищення рівня їх працездатності. До чинників естетичного впливу належить колір та музика.

Естетичне оздоблення виробничих приміщень сприяє підвищенню продуктивності праці і рівня промислової безпеки та загальному поліпшенню умов праці.

6.2. Вимоги пожежної безпеки

Пожежна безпека забезпечується шляхом проведення організаційних, технічних та інших заходів, спрямованих на попередження пожеж, забезпечення безпеки людей, зниження можливих майнових втрат і зменшення негативних екологічних наслідків у разі їх виникнення, створення умов для швидкого виклику пожежних підрозділів та успішного гасіння пожеж.

В приміщенні пожежа може виникнути внаслідок причин електричного та неелектричного характеру. Серед причин електричного характеру слід відмітити:

– коротке замикання. Струми короткого замикання та супроводжуючі

їх теплові і динамічні впливи можуть викликати руйнування електрообладнання. Профілактичними заходами в приміщенні від короткого замикання є правильний вибір провідників, деталей та апаратури, своєчасні профілактичні огляди, ремонти та тренування. Для швидкого відключення системи при виникненні короткого замикання в колах живлення вбудовані плавкі запобіжники;

– перевантаження провідників струмами, що перевищують допустимі по нормах значення. Для уникнення перевантаження підібрані правильні значення поперечного перерізу провідників та здійснюється контроль за виконанням нормативів по навантаженню, згідно зазначених в документації на обладнання;

– дія дуги та іскріння. Може виникнути в місцях підключення обладнання та механічного під'єднання струмонесучих частин. Для уникнення нещільних з'єднань під час проведення профілактичних робіт здійснюється їх перевірка.

Приміщення забезпечене протипожежним інвентарем (вуглекислотним вогнегасником типу ВВ-2 із розрахунку один вогнегасник на 40-50м2 площі приміщення). Проходи між рядами і вихід не загромаджені.

План евакуації передбачає наступні дії персоналу:

– вимкнення силової мережі пакетним вимикачем;
– закриття вентиляційного каналу заслонкою;
– евакуація працівників за схемою здійснюється на 1-му поверсі будинку.

План евакуації з ТОВ «ЛЛРЗ», м. Львів зображено на рис. 6.3.

Рис. 6.3. Схема евакуації з приміщення.

6.2. Висновки

В даному розділі передбаченні заходи по охороні праці, що відповідають вимогам нормативних документів і актів та забезпечують нормальну, ефективну і безпечну для здоров'я людини життєдіяльність на виробництві. Рішення питань штучного освітлення підтверджено відповідними розрахунками. Враховані питання пожежної безпеки.

ВИСНОВКИ

У дипломному проекті розглянуто методи й засоби розробки настільних проектів, а що дозволяють створювати зберігати та передавати інформацію.

Застосування мови програмування Java дозволило якісно виконати проект, а допоміжні мови та процедури зробили цей процес легшим та швидшим. Його основне призначення – зберігання та обробка даних про дорогоцінні камені та вироби створені з них.

В ході виконання даної кваліфікаційної роботи були отримані знання про принципи побудови і розробки баз даних і настільних додатків.

Перевага вибраної мови програмування у можливості запустити цю програму практично на будь-якому пристрої, простоті використання, великій кількості джерел інформації.

Розробка додатку може бути використана в комерційних цілях, організації обліку дорогоцінного каміння, з доопрацюванням можна додати інтеграцію для створення та візуалізації 3-х вимірних макетів.

СПИСОК ЛІТЕРАТУРИ

1. http://maven.apache.org
2. [Закон України ”Про охорону праці” – Введ. з 14.10.92. (з усіма редакціями до 2015 року).
4. Роберт Седжвік, Кевін Уейн. Алгоритми на Java, 4-е вилдання = Algorithms, 4th Edition. — М.: «Вільямс», 2013. — 848 с. — ISBN 978-5-8459-1781-2
5. Баррі Берд. Програування на Java для чайників, 3-е видання = Beginning Programming with Java For Dummies, 3rd Edition. — М.: «Діалектика», 2013. — 384 с. — ISBN 978-5-8459-1834-5.
6. Брюс Еккель. Філософія Java = Thinking in Java. — 3-е вид. — 976 с. — ISBN 5-88782-105-1.
7. Закон України ”Про охорону праці” – Введ. з 14.10.92. (з усіма редакціями до 2015 року).
8. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98, затверджені постановою Головного державного санітарного лікаря України від 10.12.1998 № 7.
9. Наказ Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду «Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин» від 26.03.2010 № 65.
10. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98, затверджені постановою Головного державного санітарного лікаря України від 10.12.1998 № 7.
11. Примірну інструкцію з охорони праці під час експлуатації електронно-обчислювальних машин, затверджену наказом Міністерства доходів і зборів України від 05.09.2013 № 443.
12. [Закон України ”Про охорону праці” – Введ. з 14.10.92. (з усіма редакціями до 2015 року).].

13. Наказ Мінсоцполітики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» від 14.02.2018 № 207 (який вступив в дію 18.05.2018 р.).
14. https://www.w3schools.com/
15. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99.
16. https://gluonhq.com/products/scene-builder/
17. https://openjfx.io/
18. https://spring.io/
19. https://github.com/yoep/spring-boot-starter-javafx
20. https://stackoverflow.com/

ДОДАТКИ

ДОДАТОК А

КОД

Клас StoneType

```
@Getter
@Setter
@Builder
@Entity
@NoArgsConstructor
@AllArgsConstructor
public class StoneType {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;
    @Column(name = "rating_value")
    private int value;
    @OneToMany(mappedBy = "type")
    private List<Stone> stones;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        StoneType stoneType = (StoneType) o;
        return id == stoneType.id && value == stoneType.value &&
Objects.equals(name, stoneType.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, name, value);
    }
}
```

Клас Necklace

```
@Getter
@Setter
@Builder
@Entity
@NoArgsConstructor
@AllArgsConstructor
public class Necklace implements GetIdAndName{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;
    private int nextStonePos = 0;
    @OneToMany(fetch = FetchType.EAGER, mappedBy = "necklace", cascade =
CascadeType.ALL, orphanRemoval = true)
    private Set<Stone> stones = new LinkedHashSet<>();
}
```

Клас Stone

```
@Getter
@Setter
@Builder
@Entity
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class Stone implements GetIdAndName{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;
    private int pricePerCarat;
    private int weight;
    double transparency;
    int posInNecklace;
    @ManyToOne
    @JoinColumn(name = "type_id")
    private StoneType type;

    @ManyToOne
    @JoinColumn(name = "necklace_id")
    private Necklace necklace;

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Stone stone = (Stone) o;
        return pricePerCarat == stone.pricePerCarat && weight == stone.weight &&
Double.compare(stone.transparency, transparency) == 0 && posInNecklace ==
stone.posInNecklace && Objects.equals(name, stone.name) && Objects.equals(type,
stone.type) && Objects.equals(necklace, stone.necklace);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, pricePerCarat, weight, transparency,
posInNecklace, type, necklace);
    }
}
```

Клас StoneTypeRepo

```
@Repository
public interface StoneTypeRepo extends JpaRepository<StoneType, Long> {
}
```

Клас NecklaceRepo

```
@Repository
public interface NecklaceRepo extends JpaRepository<Necklace, Long> {
    Necklace getNecklaceById(long id);
    Page<Necklace> findAll(Pageable pageable);
}
```

Клас StoneRepo

```
@Repository
public interface StoneRepo extends JpaRepository<Stone, Long> {
    Page<Stone> findAll(Pageable pageable);
    Page<Stone> findAllByNecklaceIsNull(Pageable pageable);
    Page<Stone>
getStoneByNecklaceAndTransparencyBetweenOrderByPosInNecklace(Necklace necklace,
double transparency, double transparency2, Pageable pageable);
    Set<Stone> getStonesByNecklace (Necklace necklace);
}
```

Файл applicatiom.yaml

```
spring:
  main:
    web-application-type: none
  profiles:
    active: h2
```

Файл applicatiom-h2.yaml

```
spring:
  datasource:
    url: jdbc:h2:mem:stones;NON_KEYWORDS=VALUE;DB_CLOSE_ON_EXIT=FALSE
    driver-class-name: org.h2.Driver
    username: test
    password:

  jpa:
    defer-datasource-initialization: true
    database-platform: org.hibernate.dialect.H2Dialect
    show-sql: true
```

Файл data.sql

```
INSERT INTO stone_type VALUES
(default,'Diamond','2'),(default,'Ruby','3'),(default,'Iolite','4')

INSERT INTO necklace VALUES (default, 'newNecklace', '0'), (default, 'necklace2',
'5')

--(id, name, pos_in_necklace, price_per_carat, transparency, weight, necklace_id,
type_id)
INSERT INTO stone VALUES (default, 'SomeDiamond',   '1', '12', '0.2', '10', '2',
'1')
INSERT INTO stone VALUES (default, 'SomeRuby',      '2', '11', '0.3', '10', '2',
'2')
INSERT INTO stone VALUES (default, 'SomeIolite',    '3', '9',  '0.1', '10', '2',
'3')
INSERT INTO stone VALUES (default, 'SecondDiamond', '4', '13', '0.7', '10', '2',
'1')
INSERT INTO stone VALUES (default, 'testIolite',    '1', '13', '0.7', '10', '1',
'1')
```

Метод openDialog

```
public void openDialog(String text){
    FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("/static/dialog.fxml"));
```

```
    fxmlLoader.setControllerFactory(applicationContext::getBean);
    data.setDialogText(text);
    Parent root;
    try {
        root = fxmlLoader.load();
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }
    Stage stage = new Stage();
    stage.setScene(new Scene(root));
    stage.setResizable(false);
    stage.show();
}
```

Метод openStonesMenu

```
public void openStonesMenu(Node node){
    data.setChooseData(ChooseData.STONES);
    openNewScene(node, StageSrc.CHOOSE_MENU);
}
```

Метод openNecklacesMenu

```
public void openNecklacesMenu(Node node){
    data.setChooseData(ChooseData.NECKLACES);
    openNewScene(node, StageSrc.CHOOSE_MENU);
}
```

Метод matchesDouble

```
public static String matchesDouble(String newValue, TextField node) {
    if (!newValue.matches("[0-1](\\.[0-9]*)?")) {
        node.setText(newValue.replaceAll("[^\\d.]", ""));
        StringBuilder aus = new StringBuilder(newValue);
        boolean firstPointFound = false;
        for (int i = 0; i < aus.length(); i++){
            if(aus.charAt(i) == '.') {
                if(!firstPointFound)
                    firstPointFound = true;
                else
                    aus.deleteCharAt(i);
            }
        }
        newValue = aus.toString();
    }
    return newValue;
}
```

Метод openDialog

```
public void openDialog(String text){
    FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("/static/dialog.fxml"));
    fxmlLoader.setControllerFactory(applicationContext::getBean);
    data.setDialogText(text);
    Parent root;
    try {
        root = fxmlLoader.load();
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }
    Stage stage = new Stage();
    stage.setScene(new Scene(root));
```

```
    stage.setResizable(false);
    stage.show();
}
```

Клас NecklaceController

```
@Component
public class NecklaceController extends SceneOpener {

    @Autowired
    private NecklaceRepo necklaceRepo;
    @Autowired
    private StoneRepo stoneRepo;

    @Autowired
    private StoneTypeRepo stoneTypeRepo;
@FXML
void initialize() {
    bound1.setText("0");
    bound2.setText("1");

    bound1.textProperty().addListener(doubleInputWithRemap(bound1));
    bound2.textProperty().addListener(doubleInputWithRemap(bound2));

    necklace = necklaceRepo.getNecklaceById(data.getCurrentNecklace().getId());
    remapButtons();
    title.setText(necklace.getName());
    valueText.setText(String.format("total value: %s",getTotalValue()));
    weightText.setText(String.format("total weight: %s", getTotalWeight()));

    nextPageButton.setOnAction(actionEvent -> {
        page++;
        remapButtons();
    });
    previousPageButton.setOnAction(actionEvent -> {
        page--;
        remapButtons();
    });

    newStoneButton.setOnAction(actionEvent ->
openEditNecklaceMenu(newStoneButton));
    backButton.setOnAction(actionEvent -> openNecklacesMenu(backButton));
}
...
switch (data.getStoneAction()){
    case VIEW -> viewStone();
    case IN_NECKLACE -> stoneInNecklace();
    case ADD_TO_NECKLACE -> addingToNecklace();
}
```

ДОДАТОК Б

ГРАФІЧНІ МАТЕРІАЛИ

Аркуш 1. Схема роботи системи.

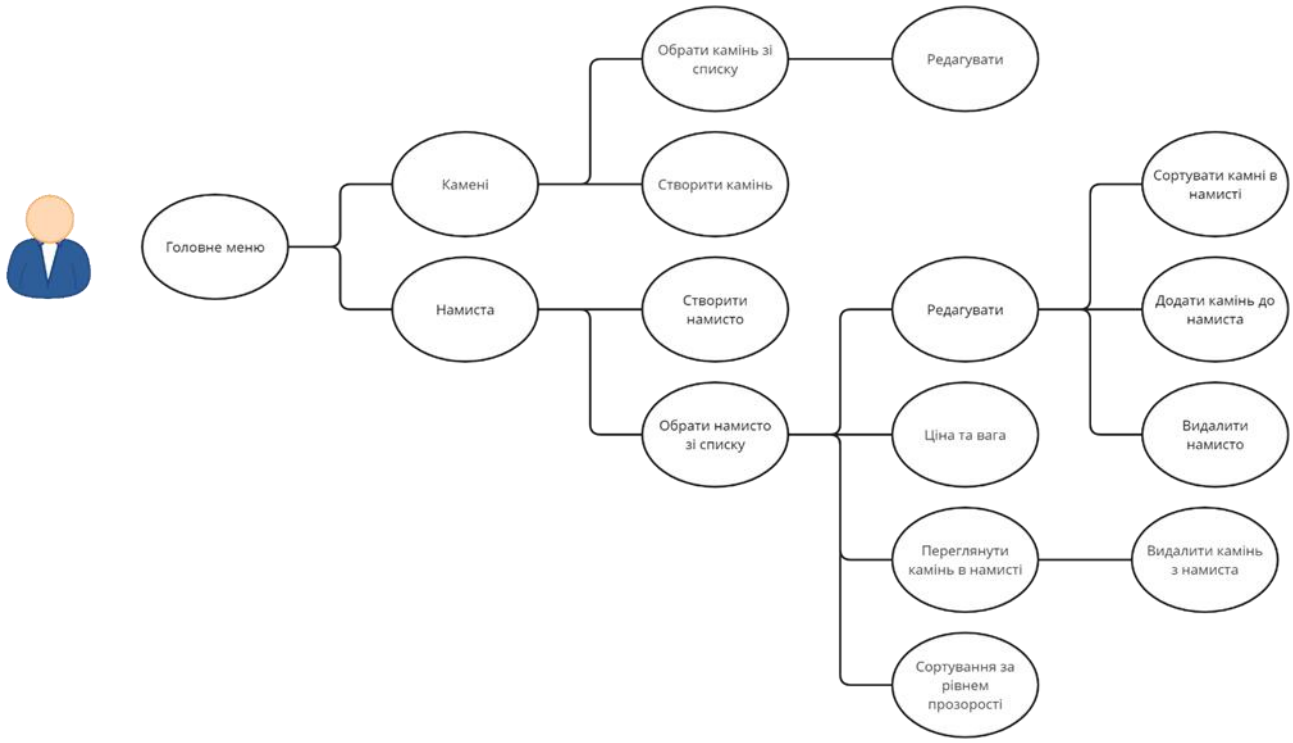

Рис Д.1. Схема роботи програми.

Аркуш 2. Вигляд інтерфейсу програми.

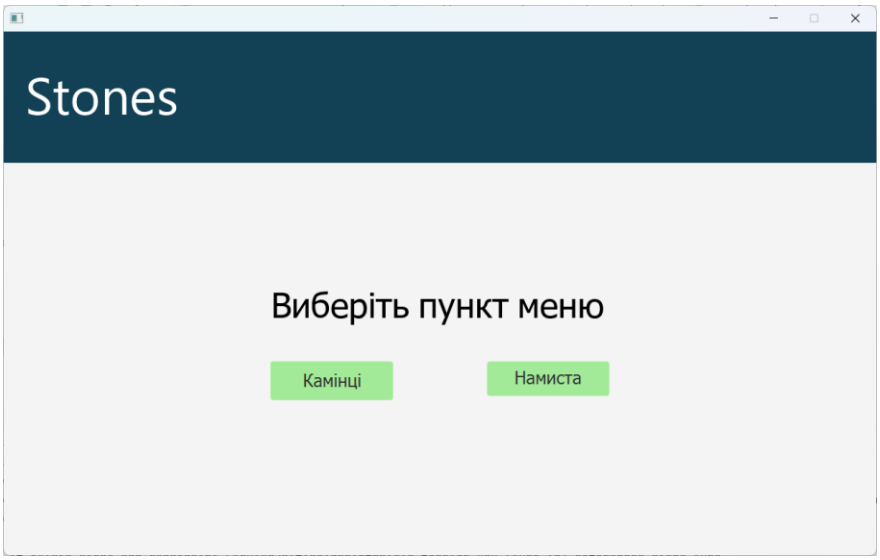

Рис Д.2.1. Головне меню.

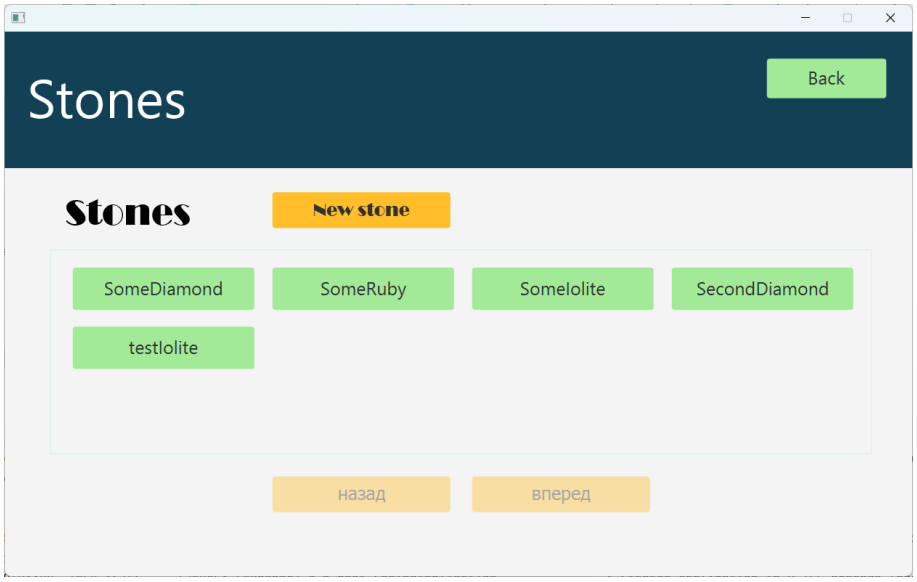

Рис Д.2.2. Камені.

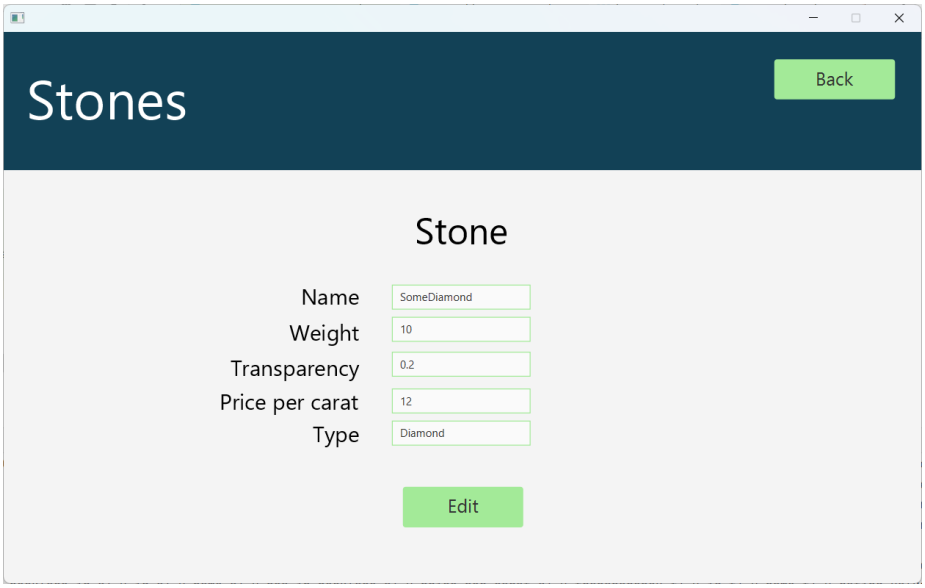

Рис Д.2.3. Камінь.

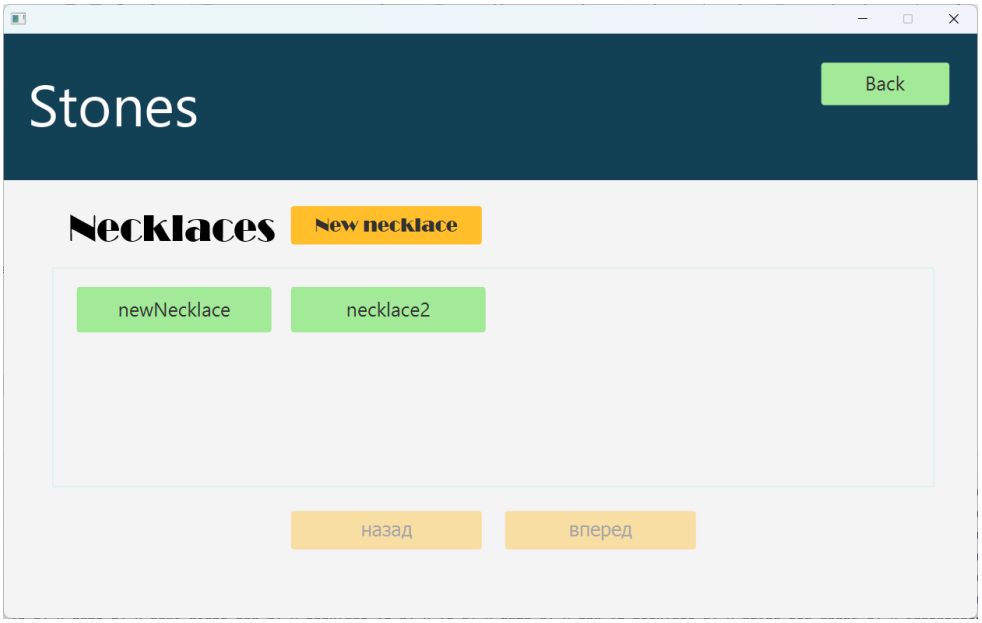

Рис Д.2.4. Намиста.

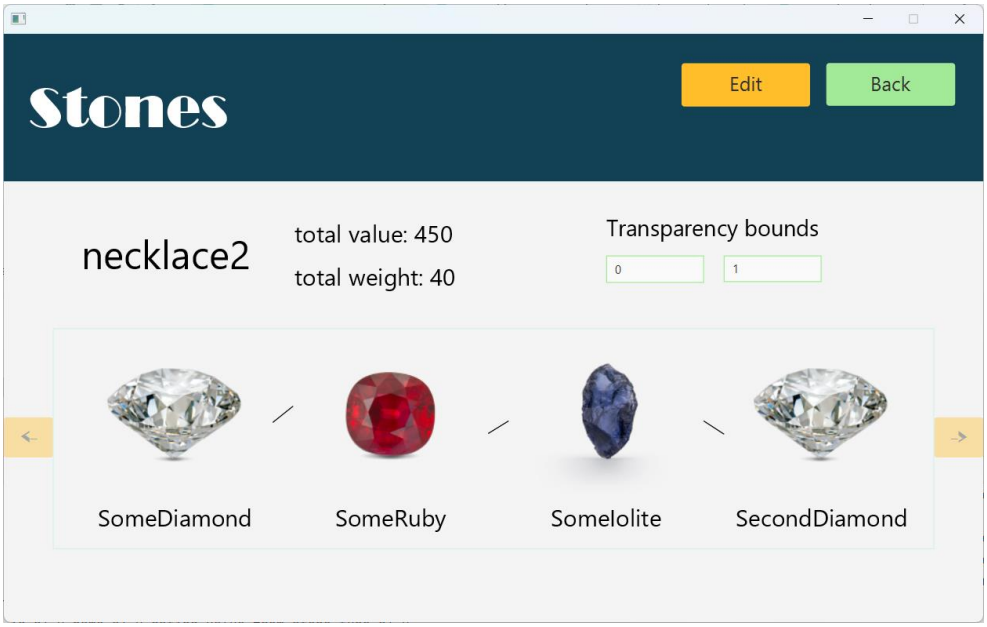

Рис Д.2.5. Намисто.

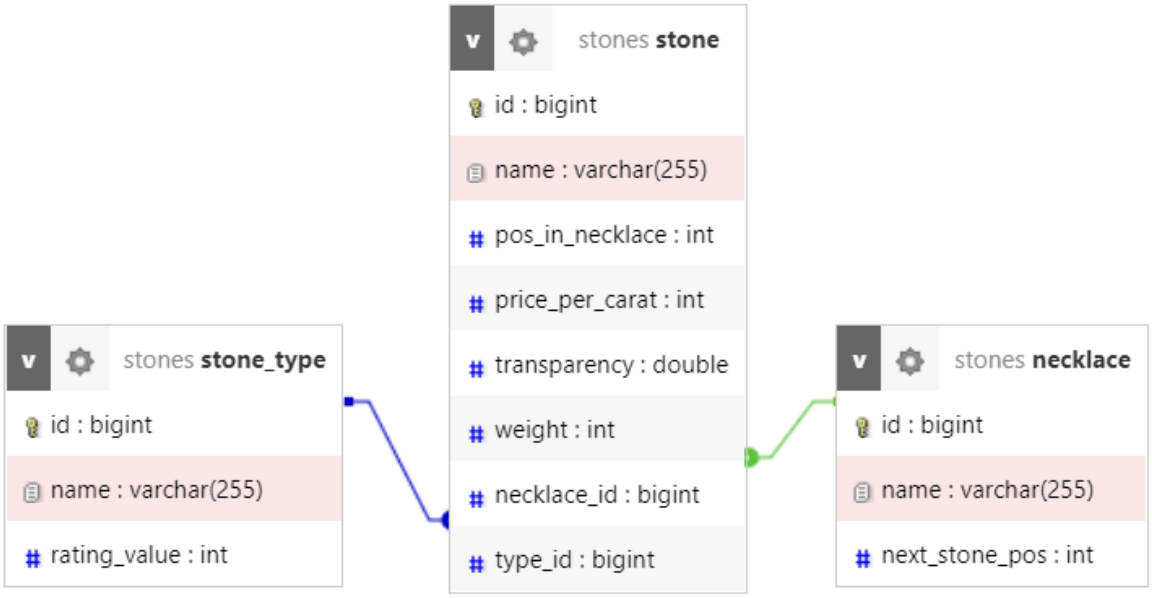

Рис Д.3. Схема бази даних.