Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

# МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

**Розробка методу та програмного засобу діагностики знань з автоматичним коригуванням рівня складності запитань**

Виконав: студент 2-го курсу, групи 1ПІ-20м
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Струбчевський А.Г.
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри ПЗ
Бабюк Н.П.
(прізвище та ініціали)

«_____»_____2021 р.

Рецензент: к.т.н., доцент кафедри КН
Крилик Л.В.
(прізвище та ініціали)

«_____»_____2021 р.

**Допущено до захисту**
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О.Н.
«____»_____2021 р.

Вінниця ВНТУ - 2021 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
« 13 » вересня 2021 р.

**З А В Д А Н Н Я**
**НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

Струбчевському Артему Геннадійовичу

1. Тема роботи – розробка методу та програмного засобу діагностики знань з автоматичним коригуванням рівня складності запитань.
Керівник роботи: Бабюк Наталя Петрівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 24 » вересня 2021 р. № 277.

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: мова програмування – Java; середовище виконання – Intellij Idea; тип обробки даних – текстовий.

4. Зміст розрахунково-пояснювальної записки: обґрунтування доцільності розробки; варіантний аналіз та обґрунтування вибору засобів реалізації додатку; розробка методу та структури програмного засобу для реалізації системи тестування; тестування роботи додатку; економічна частина.

5. Перелік графічного матеріалу: мета роботи, об'єкт та предмет дослідження; основні задачі дослідження; порівняльний аналіз аналогів; наукова новизна, практичне значення; розробка методів; приклад вигляду розробленої програми; засоби реалізації; висновки, публікації, апробації.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-4 | Бабюк Н.П., к.т.н., доц. каф. ПЗ | 14.09.2020 р. | 01.12.2020 р. |
| 5 | Ратушняк О.Г., к.е.н, доцент кафедри ЕПВМ | 14.09.2020 р. | 28.11.2020 р. |

7 Дата видачі завдання 14.09.2021 р.

8. Календарний план.

| № | Назва етапів виконання магістерської роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Постановка задачі роботи | 14.09.21 | |
| 2 | Огляд існуючих методів та систем тестування | 14.09-17.09.21 | |
| 3 | Аналіз та вибір методів тестування | 17.09-25.09.21 | |
| 4 | Розробка серверної частини додатку | 25.09-13.10.21 | |
| 5 | Розробка алгоритму автоматичного коригування рівня складності запитань | 14.10-22.10.21 | |
| 6 | Розробка інтерфейсу та тестування програми | 22.10-01.1.21 | |
| 8 | Підготовка матеріалів та опис розробки систем тестування | 02.11-15.11.21 | |
| 9 | Розрахунок економічної частини роботи | 16.11-01.12.21 | |
| 10 | Оформлення пояснювальної записки та ілюстративного матеріалу | 02.12-07.12.21 | |
| 11 | Аналіз виконання роботи, висновки, додатки | 08.12-06.12.21 | |
| 12 | Перевірка якості виконання магістерської роботи та усунення недоліків | 13.12.21 | |

**Студент** _____ **Струбчевський А. Г.**

( підпис ) (прізвище та ініціали)

**Керівник магістерської кваліфікаційної роботи** _____ **Бабюк Н. П.**

( підпис ) (прізвище та ініціали)

# АНОТАЦІЯ

УДК 004.04

Струбчевський А.Г. Розробка методу та програмного засобу діагностики знань з автоматичним коригуванням рівня складності запитань. Магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2021. 130 с. На укр. мові. Бібліогр.: 31 назв; рис.: 34; табл. 15.

Магістерська кваліфікаційна робота присвячена розробці програмного засобу для створення індивідуальної системи тестування з автоматичним коригуванням рівня складності запитань. Для вирішення поставленої задачі було використано такі мови програмування як C# і Java, а також інструмент для створення інтерфейсу Windows Forms. В якості бази даних була використана PostgreSQL.

Розроблено метод реалізації автоматичного рівня коригування складності запитань під час проходження тестування для більш точного визначення рівня знань.

Розроблено програмний засіб EasyQuiz який повинен задовольнити потребу користувачів зокрема керівників або викладачів у забезпеченні простого та швидкого способу створення системи контролю знань або навичок. При використані даного додатку користувач зможе швидко та максимально зручно провести тестування, а якщо необхідно внести автоматичні коригування до складності тестів, завдяки зручній структурі навігації та функціоналу для створення системи тестування.

В магістерській роботі були також виконані економічні розрахунки по визначенню доцільності нового програмного продукту.

# ANNOTATION

UDC 004.05

Strubchevsky AG Development of a method and software for knowledge diagnostics with automatic adjustment of the level of complexity of questions. Master's thesis in specialty 121 - Software Engineering. Vinnytsia: VNTU, 2021. 130 p. In Ukrainian language. Bibliogr .: 31 titles; fig .: 34; table 15.

The master's qualification work is devoted to the development of software for creating an individual testing system with automatic adjustment of the level of complexity of questions. To solve this problem, programming languages such as C # and Java were used, as well as a tool for creating the Windows Forms interface. PostgreSQL was used as the database.

The method of realization of automatic level of adjustment of complexity of questions during passing of test for more exact definition of level of knowledge is developed.

EasyQuiz software has been developed to meet the needs of users, including managers or teachers, in providing a simple and fast way to create a system of control of knowledge or skills. When using this application, the user will be able to quickly and easily perform testing, and if necessary, make automatic adjustments to the complexity of the tests, thanks to the convenient navigation structure and functionality to create a testing system.

In the master's thesis, economic calculations were also performed to determine the feasibility of a new software product.

ЗМІСТ

# ВСТУП

Завдяки постійному розвитку, необхідно вивчати все більшу кількість нової інформації, що відповідно потребує все більше засобів для її контролю та перевірки. Для того, щоб структурувати процес засвоєння матеріалу під час навчання або роботи, її можна також перевірити за допомогою класичного методу проходження письмових тестів або ж проведення співбесіди, але для класичних методів необхідні затрати людських ресурсів в значній мірі адже кожному потрібно виділити час для проведення контролю.

На сьогоднішній день існує багато систем електронного тестування але кожна система розроблена під свій стиль та галузь використання, а також має певний набір базовий функцій. Тести повинні бути створені таким чином, що питання, умови проведення і оцінювання заздалегідь погоджені, процеси проведення і оцінювання визначені наперед [1].

Правильна форма подання текстових або графічних тестових завдань [2]:

− тестові завдання однакової форми супроводжуються однією інструкцією для їх виконання (при зміні форми - змінюється інструкція);

− текст інструкції відрізнятися від основного тексту (шрифтом, кольором) та відокремлюватися від тестових завдань двокрапкою;

− завдання нумеруються арабськими цифрами, нумерація завдань різної форми наскрізна;

− запитальна частина завдання формулюється у стверджувальній формі стисло, чітко, без подвійного тлумачення та виділяється великими літерами або активним кольором;

− запитальна частина тестових завдань та можливі відповіді не відокремлюються будь-яким знаком;

− елементи відповіді частини тестового завдання мають окрему індексацію;

− відповіді розміщуються під запитальною частиною симетрично або поряд з нею;

− якщо відповідь передбачає певну процедуру обчислення, то остання має бути простою, без потреби застосування складних технічних засобів.

− форма подання тестових завдань не змінюється в межах блоку завдань, призначеного для тестування.

**Актуальність дослідження** полягає в тому, що існує необхідність сучасного безкоштовного додатку, що матиме інтуїтивно зрозумілий та багатомовний інтерфейс та можливість корегування складності завдань для кращого розуміння рівня знань опитуваних користувачів.

**Об'єктом дослідження** є процес створення та автоматизації системи індивідуального середовища тестування для контролю знань.

**Предметом дослідження** є методи створення систем тестування а також алгоритми застосовані в автоматичних системах тестування для більш точного визначення рівня знань користувача.

**Мета роботи —** полягає в підвищенні якості тестування за рахунок розробки простої тестової системи з використанням покращеного методу автоматичного коригування рівня складності запитань, що забезпечить можливість покращення якості визначення рівня знань користувачів.

Відповідно до поставленої мети в дипломному проекті потрібно вирішити такі завдання:

- проаналізувати існуючі системи і засоби тестування
- виявити основні недоліки в існуючих аналогів та врахувати їх для створення власного додатку
- розробити основний алгоритм системи тестування;
- розробити сучасний дизайн, що дасть змогу орієнтуватись в ньому без додаткового навчання;
- створити просту універсальну платформу тестування;
- розробити алгоритм автоматичного коригування рівня складності запитань
- провести тестування програмного продукту для перевірки всіх можливих варіантів використання.

**Зв’язок роботи з науковими програмами, планами, темами.** Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

**Наукова новизна отриманих результатів -** модифіковано метод автоматичного коригування рівня складності запитань для створення системи тестувань, що дозволило, порівняно із наявними методами, пришвидшити час тестування, оскільки ефективніше аналізується рівень знань користувачів.

**Практична цінність роботи**. Практична цінність полягає в розробці програмного додатку, для створення системи тестування для комп’ютерних систем з алгоритмом автоматичного коригування рівня складності запитань, що підвищує точність оцінювання на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень.

**Методи дослідження.** У процесі досліджень магістерської роботи було проведено аналіз завдання, а також проведення оцінки існуючих аналогів, також методики існуючих методів тестування та способів їх покращити, після чого було прийнято рішення про розробку додатку для створення системи тестування, який було створено у відповідності поставленим функціональним задачам.

**Апробація матеріалів магістерської кваліфікаційної роботи**. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на конференції «Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії» (Вінниця, 2019), а також на конференції «Інформаційні технології та автоматизація» (Одеса, 2021).

**Публікації.** Основні результати досліджень опубліковано в 2 наукових працях.

Вимоги до програмного додатку були виконані в повному обсязі, що дає можливість використання його як повноцінну систему тестування та впроваджувати на підприємства та навчальні заклади.

# 1 АНАЛІЗ ЗАВДАННЯ І МЕТОД ЙОГО ВИРІШЕННЯ

## 1.1 Аналіз стану систем тестування

Через великий об’єм інформації який необхідно опрацьовувати людям щоденно, незалежно від того чи ти ще навчаєшся чи працюєш виникла необхідність створення систем контролю якістю знань, а також їх структуризації ти створення чіткої ієрархії рівня знань та навичок у суспільстві. Перевірка знань очікує нас у всіх сферах життя - наприклад, коли роботодавець цікавиться наскільки добре ти знаєш свій процес, чи має людина достатню кількість знать для підвищення заробітної плати, чи коли викладач перевіряє як добре ти засвоїв матеріал дисципліни, для того щоб дати питання на всі ці відповіді потрібно провести тестування.

Методи контролю знань ви можете побачити на рисунку 1.1

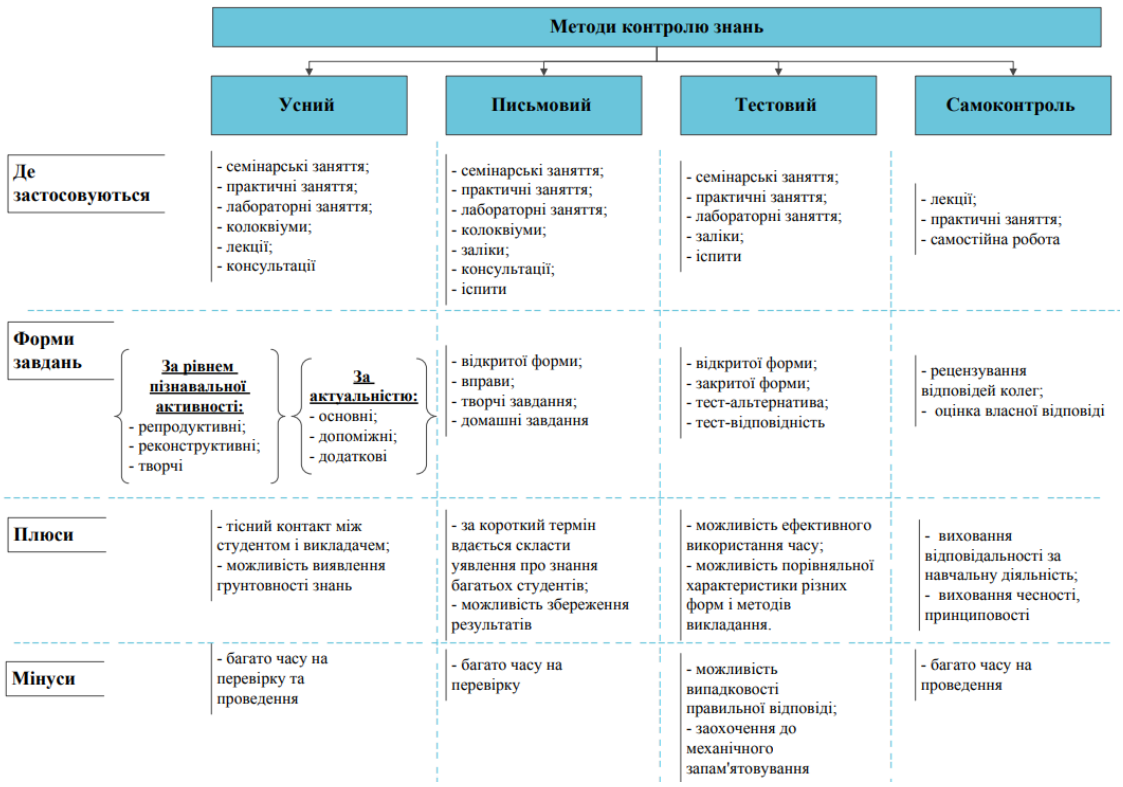



Рисунок 1.1 – Методи контролю знань

До розгляду візьмемо приклад, скільки тестувань відбувається під час одного тільки навчального процесу.

Самоконтроль знань – найпростіший вид контролю, що передбачає надання студентам запитань і завдань, на які він повинен відповісти самостійно. У випадку ускладнень кожен студент має можливість звернутися до матеріалів підручника для пошуку відповіді на поставлені питання. Основною метою проведення самоконтролю вбачається самоствердження, досягнення впевненості студента у рівні отриманих знань (це може бути суб'єктивним баченням кожного як особистості).

Вхідний контроль, його проведення переслідує кілька цілей, які залежать, в свою чергу, від цілей навчальної дисципліни і її специфіки. По-перше, за його результатами можна визначати рівень підготовленості кожного студента до вивчення дисципліни ("на вході"). Тобто проведення вхідного контролю слугує виконує роль специфічного допуску кожного студента до навчальної діяльності. У цьому випадку вхідний контроль розглядається як констатувальний. По-друге, можна організувати вхідний контроль, надаючи йому діагностично-коригувальної функції. За результатами виконання тестових завдань розкриваються прогалини у знаннях студентів, які необхідно компенсувати до початку вивчення дисципліни.

Таким чином, навчальний курс стає адаптивним – кожен студент рухається своїм шляхом, в залежності від рівня початкової підготовки. Додатковий ефект вихідного контролю розкривається у формуванні побічної функції: виконання тестових завдань сприяє налаштуванню студентів на відповідну предметну галузь, вводить в специфічну термінологію, сприяє актуалізації необхідних знань та стає своєрідним стартовою платформою для початку навчальної діяльності. Це забезпечує успішність навчально-виховного процесу. Зазначене підтверджується досвідом проведення "вхідного тестування студентів першого курсу напряму підготовки 6.020303 "Філологія" (мова і література) факультетів германської філології, романської філології, сходознавства. Тестові завдання були спрямовані на виявлення

рівня отриманих знань та сформованих умінь та навичок у процесі вивчення предмету "Інформатика" в ЗНЗ. Аналіз отриманих результатів дозволив з'ясувати необхідність виокремлення у навчальній програмі ВНЗ коректувального етапу навчання, спрямованого на заповнення виявлених прогалин у знаннях студентів.

Поточний контроль, ціллю його проведення є діагностика ЗУН у процесі засвоєння навчальної теми і, за необхідності, здійснення корекції навчання – діагностично-коригувальна функція. Систематичне проведення контролю поточного рівня засвоєння ЗУН дозволяє коректувати хиби у навчанні й досягати необхідного рівня засвоєння навчального матеріалу.

Рубіжний контроль здійснюється з метою визначення рівня засвоєння чергового розділу (теми) дисципліни. Студентам може пропонуватися завдання творчого характеру, підвищеної складності або завдання, яким передбачається перенесення засвоєних знань на інший матеріал. Їх успішне виконання стає свідченням того, студент опанував всю систему ЗУН, передбачених цілями даної теми. За проведення рубіжного контролю студент може використовувати допомогу у вигляді необхідного довідкового або інформаційного матеріалу, порад, роз'яснення помилок тощо. Також, рубіжний контроль може слугувати в якості своєрідного вхідного контролю, який проводиться з метою допуску до вивчення наступного матеріалу і підтримки рівня знань за умови великих перерв у навчальній діяльності роботі.

Заключний (підсумковий) контроль застосовується для отримання результаті, які отримані по закінченню навчання. Він включає серію завдань, що охоплюють весь діапазон вивченого матеріалу, всі завдання кожен студент повинен виконати самостійно. За результатами підсумкового контролю, як правило, викладач здійснює оцінювання успішності [2].

І це лише невеличка частинка усього процесу тестування і розглянута лише одна сфера в соціумі, можна тільки уявити скільки в день проводиться тестувань, кожне з яких відповідає своїй галузі, тому створення додатків для контролю знань повинно спростити даний процес та пришвидшити його.

Створення рукописної бази для такого роду тестування вимагає велику кількість часу, адже необхідно визначити структуру роду тестування чи це будуть тести звичайного вигляду, де є стандартне питання і вибір між 4 відповідями, або варіант із відкритою, повною, розгорнутою відповіддю або ж інші методи. Також самі бланки тестів повинні бути надруковані та роздані суб'єктам, для об'єктивного тестування потрібно обмежити їх можливість доступу до інших джерел інформації, що також вимагає людських затрат як і їх відповідна перевірка. Під час такого методу можливе збільшення такого поняття як людський фактор, що також може вплинути на результати дослідження.

На даний момент існує багато варіантів проведення тестування, класичні за допомогою ручки та папірця, метод дискусії, інтерв'ю, але найбільш зараз стало популярним проведення електронних тестувань.

## 1.2 Аналіз методів вирішення поставленої задачі

Як метод вирішення завдання використовується системи електронного тестування. Пояснити таку популярність таких систем досить легко відповідно до вище поданого, оскільки у них є цілий ряд незаперечних переваг. що як на мене мають певний ряд переваг:

- Не потрібно довго перевіряти відповіді що значно економить час людини, що проводить тестування;
- Всі дані зручно автоматизовані в базі даних користувача
- Потрібно розробити дану систему тільки раз, а далі лише надавати доступ протягом тривалого часу
- Легкість змін та модифікації системи

І це ще далеко не всі переваги даного методу тестування [3].

Такий метод тестування може стати у пригоді керівнику та педагогічному колективу навчальних закладів для різних форм наукової, методичної та організаційної роботи, може використовуватись як і в ігровій

формі так і більш офіційній для проведення контролю на підприємствах, фірмах, офісах [4].

Поява різних систем тестування пов'язана із швидким наростанням темпів необхідної бази знань для всіх галузей діяльності людини. Також як було сказано вище, необхідність контролю цих знань збільшується відповідно. Така практика електронного тестування допомагає зменшити кількість часу, необхідного для отримання результатів і прийняття згідно них відповідних рішень, оцінок, балів тощо, оскільки зникає необхідність у тривалих і втомливих досліджень, на створення певної тестової структури, достатньо просто сісти за комп'ютер і використати весь базовий інструментарій наданий програмою для задання необхідних питань. На відміну від паперових варіантів, адже за допомогою легкого доступу та миттєвого редагування, дублювання значно спрощує саму підготовку до процедури, а видача інформації одразу ж після дає змогу миттєво її обробити. Саме тому є нагальна потреба в створенні програмного додатку EasyQuiz з можливістю мультикористувацього доступу до неї.

Але хоч з створенням додатку проведення тестування значно спрощується необхідно дотримуватись певних норм під час процесу конструювання питань тесту працівнику необхідно дотримуватися системи методологічних вимог і принципів, а саме:

− неправильні відповіді необхідно підбирати відповідно до типових помилок;

− формулювання питання має відрізнятися від формулювання визначень у навчальних матеріалах;

− правильні відповіді до тестів мають мати різні позиції;

− відповіді на питання мають виглядати логічно та правдоподібно, змушуючи студентів аналізувати кожен варіант відповіді та виявляти неточності;

− під час конструювання тестів необхідно уникати повторів, а відповіді не повинні бути підказками до інших питань тесту;

− питання, що підлягають оцінці, мають бути настільки широкі, щоб можна було охопити ключові моменти теми, яка підлягає вивченню;

− необхідно наводити декілька правильних відповідей, кожна з яких доповнює інші правильні відповіді;

− студенти мають знати зміст, терміни та тривалість контролю;

− тестовий контроль має активізувати творче та свідоме ставлення студентів до навчання, стимулювати зростання пізнавальних потреб, інтересів, а також організовувати навчальну діяльність;

− варіанти відповідей розрахункових завдань мають містити не випадкові значення, а лише ті, які було отримано під час розв'язування завдання з урахуванням типових помилок, адже це мінімізує випадковість.

Також в розробленому додатку є можливість автоматичного коригування рівня складності запитань, коли система бачить що людина що тестується відповідає не вірно не більшість запитань вона автоматично вибирає питання що мають нижчий рівень складності та можливо досягнути більш точнішого визначення рівня знань, хоча в додатку присутній і звичайний метод тестування з вимкненим алгоритмом.

Розглянемо існуючи види тестування. Дидактичний тест є множиною певних тестових завдань, яка надає можливість якісно виміряти рівень засвоєння знань та навичок суб'єктів що тестуються.

Касифікація типів тестових завдань наразі повністю не визначена. Існують різні системи розподілу на класифікації. В залежності від критерію, одержують різні типи тестових завдань. За способом виконання тестові завдання поділяють на закриті та відкриті.

– логічний тип – відповідь на питання суб'єкт вибирає між двома варіантами;

– множинний вибір – суб'єкт обирає відповідь на питання з декількох варіантів, причому питання можуть мати декілька правильних відповідей;

– коротка відповідь – відповіддю на питання є слово або коротка

фраза, яку вводять в текстовому варіанті, при цьому допускається кілька правильних відповідей;

– відповідність – кожному елементу першої групи потрібно зіставити елемент який відповідає до іншої другої групи;

– числовий – відповіддю на питання про виконання обчислювальних операцій є число, при цьому значення може мати інтервал гранично допустимої похибки відхилення;

– есе – студент коротко викладає свої думки на запропоновану проблему;

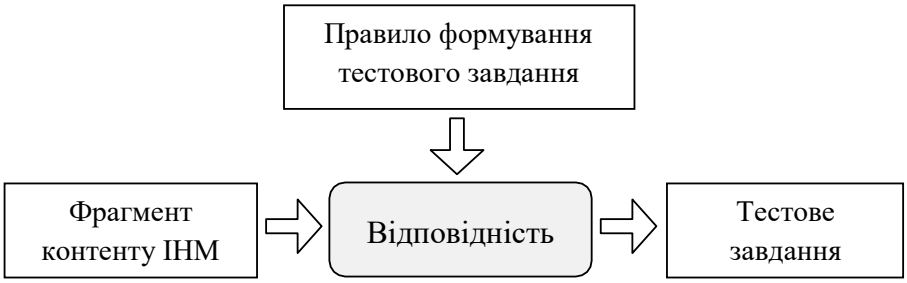



Рисунок 1.2 – Правила продукції для сформування тестового завдання

З наведеного можна зробити висновок, що процес ручного створення тестів базується у використанні заснованої на правилах моделі, і дозволяє подавати знання у вигляді конструкцій типу (рисунок 1.2). Таку модель використовують продукційні системи, що працюють аналогічним чином для вирішення практичних задач.

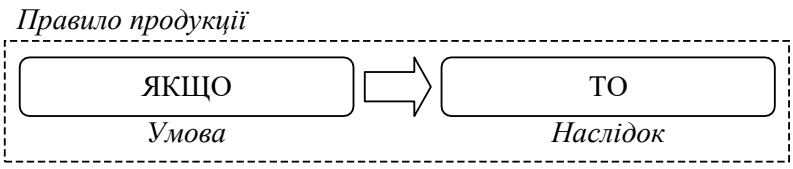



Рисунок 1.3 – Логічна структура правила продукції

У продукційних системах знання подані у формі множини правил продукції, за якими сформовуються висновки, що повинні бути записані в різних ситуаціях. Висновки робляться за методами прямого та зворотного

логічного висновку. Розрізняють два види продукційних систем залежно від методу логічного висновку системи: з прямим логічним висновком та системи із зворотним логічним висновком .

Загальна стратегія вирішення задач визначається в розбитті їх на фрагменти. Системи з прямим логічним висновком знаходяться за управлінням фактів. Починаючи свою роботу з відомих фактів і продовжують, використовуючи правила для створення висновків або ж виконання певних дій. Системи з зворотним логічним висновком зазвичай керуються гіпотезами. Вони починають свою роботу з гіпотези, яку користувач намагається довести і відшукують правила, які дозволять довести коректність цієї гіпотези .

У випадку с процесом створення тестів, прямий логічний висновок відповідає створенню нових тестових завдань за правилами; зворотний логічний висновок відповідає пошуку правил, що були використані при перетворенні фрагменту існуючого контенту ІНМ в елемент тестового завдання.

З наведеного можна зробити висновок, що використання продукційної моделі для подання алгоритмів створення тестових завдань відповідає процесу ручного створення тестів, який характеризується циклічним повторенням однотипного процесу, в якому кожна ітерація циклу відповідає вирішенню задачі створення одного тестового завдання. Формалізація процесу ручного створення тестових завдань за використання заснованої на правилах моделі знань відкриває можливості для його автоматизації.

Адаптивним називається тест, в якому складність чи інші властивості тестових завдань змінюються залежно від правильності відповідей випробуваного. Наприклад, якщо студент правильно відповідає на тестові завдання, складність наступних завдань підвищується, якщо неправильно – знижується. Також є можливість внесення додаткових питань в області, яку особа знає не дуже добре для більш тонкого з’ясування рівня знань у цих галузях. Дана модель застосовується для тестування студентів за допомогою комп’ютерних засобів, тому що «вручну» неможливо заздалегідь розмістити

стільки питань і в тому порядку, скільки і в якому вони повинні бути пред’явлені особі, що тестується. Системи адаптивного тестування визнаються більш ефективними за існуючі класичні системи.

Відповідно до існуючих визначень, ефективною вважається така система тестування, у якій навчання й контроль знань проходять за мінімально можливий час, причому забезпечується їхня повнота, а також відсутність або мінімальна присутність інформаційної надмірності або недостатності і максимально можлива об’єктивізація отриманих результатів. Відомо, що чисельним вираженням ефективності є критерій ефективності. Рішення завдання пошуку максимального значення критерію ефективності пов'язане з формалізацією й оптимізацією зазначених вище понять.

Під час проектування контролю знань, потрібне графоподібне подання навчального матеріалу та множини різних тестових завдань. Кожний з рівнів навчального матеріалу може містити ще ряд підрівнів. Існує також ще й вертикальний поділ, за допомогою якого найчастіше виділяються теми НК. В процесі тестування бажаним є підхід, коли для переходу на наступний рівень потрібно зважувати результат від відповідей на питання попереднього рівня, що і є одним зі способів усунення інформаційної недостатності. Інформаційна надмірність виникає під час неправильної реалізації структури контролю знань.

Наступні вид це адаптивне тестування.

– На початку тестування видав завдання середньої складності й потім, кожному визначається завдання складніше або легше.

– Контроль знань відбуваєьбся з будь-якого рівня складності завдань і далі ітераційно наближається до рівня складності, що відповідає рівню знань.

– База знань, з питаннями, розподіленими по рівнях складності. При правильній відповіді завдання обирається із більш складних завдань, при неправильному – з більш легких.

– Тестування розпочинається з перевірки знання терміна та у

випадку коректних відповідей підвищуєть складність методом перевірки знання менш важливих термінів.

Підводячи висновки з вище описаного, обов’язковою вимогою до автоматизованого створення множин тестових завдань визначено одночасне формування необхідних зв’язків тестового завдання з семантичними і структурними елементами. Це дозволить використовувати їх для тестування не тільки з традиційним алгоритмом, а й для тестування використовуючи технології адаптивного тестування, що відповідає сучасним методам в розвитку індивідуальних систем тестування.

Проаналізовані існуючі методи автоматизації формування тестових завдань переважно вимагають тривалої та трудомісткої попередньої підготовки.

## 1.3 Порівняльний аналіз аналогів

У ході дослідження можливих шляхів вирішення поставленої проблеми, були знайдені основні аналоги даного програмного продукту. Методом структурування та пошуку були виявлені головні недоліки даних аналогів, а також означено переваги розроблюваного додатку, що надає можливість визначити основні ідеї для його створення та визначення основних напрямків над якими була проведена робота по виправленню недоліків в власній розробленій структурі для індивідуального тестування. Результати порівняльного аналізу наведені в табл. 1.1

Kahoot – платформа для створення вікторин, тестів, дидактичних ігор.

Сервіс може бути використано для перевірки знань учнів [5].

ISpring – iSpring Suite допомагає автоматизувати оцінку знаній в компанії. Коли необхідно провести атестацію і отримати об'єктивну інформацію про рівню підготовки співробітників в всіх філіалах.[6].

QuizMaker – зручний веб-сервіс для створення та проходження тестових систем [7].

Таблиця 1.1 – Порівняльний аналіз додатків Kahoot, ISpring, QuizMaker та EasyQuiz.

| Критерії | Kahoot | ISpring | QuizMaker | EasyQuiz |
|---|---|---|---|---|
| Сервіс може бути використано для перевірки знань учнів | + | – | + | + |
| Можливість автоматизувати оцінку знань в компанії а також автоматичне коригування рівня складності запитань | – | – | – | + |
| Портативність | + | + | − | + |
| Безкоштовний | + | − | + | + |
| Простота користування | + | − | + | + |
| Всього | 4 | 1 | 3 | 5 |

Перший і другий критерій який був вибраний для порівняння і показує наскільки додаток є мультиорієнтованим і чи може використовуватись в різних галузях. Можливість використовувати додаток для контролю знань учнів, студентів в освітній галузі або застосування його для перевірки наявності інформації необхідної для виконання обов’язків тієї чи іншої посади відповідно.

Третій критерій пов’язаний з легкістю поширення та власної малої ваги програми, що займає невелику частину процесорного часу, об’єму оперативної пам’яті та жорсткого диску.

Четвертий критерій розгортає питання поширення додатку на платній або безкоштовній основі, чи обмежується функціональність платформи і необхідність придбання преміум версії програми.

П’ятий і останній критерій показує наскільки інтуїтивним є інтерфейс. Простота створення системи тестування, доступу до неї та виведення результатів.

В результаті проведеного аналізу, можна стверджувати важливість та необхідність обрання даної теми для подальшої розробки програмного додатку і вирішення поставленої проблеми.

## 1.4 Постановка задачі

Дипломний проект присвячено розробці додатку для створення індивідуальної системи тестування: його внутрішньої складової та зовнішнього вигляду. Для зовнішнього вигляду додатку (його інтерфейсу) було висунуто наступні вимоги:

- проаналізувати існуючі системи і засоби тестування
- виявити основні недоліки в існуючих аналогів та врахувати їх для створення власного додатку
- розробити основний алгоритм системи тестування;
- розробити сучасний дизайн, що дасть змогу орієнтуватись в ньому без додаткового навчання;
- створити просту універсальну платформу тестування;
- розробити алгоритм автоматичного коригування рівня складності запитань
- провести тестування програмного продукту для перевірки всіх можливих варіантів використання.

Для того щоб виконати поставлені вимоги потрібно використати ряд сучасних технологій:

- Windows Forms інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework [8].

- систему керування базами даних (PostgreSQL) – вільна об’єктно-реляційна система керування базами даних [9].
- мову програмування C# – що надає можливість роботи нашого додатку з інтерфейсом.
- технологію Spring та мову Java – для взаємодії із серверною частиною програми, що відповідає за збереження та завантаження інформації[10].

## 1.5 Висновок

Було розглянуто та проаналізовано стан систем індивідуального тестування, що відповіло на питання актуальності даної розробки. Аналіз методів вирішення та порівняння аналогів: Kahoot, ISpring, QuizMaker, EasyQuiz. Дозволило пояснити необхідність, виправлення помилок та покращення уже існуючих систем та встановити задачу яку необхідно було виконати.

# 2 РОЗРОБКА СТРУКТУРИ І АЛГОРИТМІВ РОБОТИ ПРОГРАМИ

## 2.1 Розробка інтерфейсу програмування Windows Forms

Щоб створити GUI програми в Microsoft .NET, потрібно використовувати Windows Forms. Windows Forms – новий стиль побудови програми на базі класів .NET Framework class library. Вони мають власну модель програмування, яка більш досконала, ніж моделі, що базуються на Win32 API або MFC, і вони виконуються в керованому середовищі .NET Common Language Runtime (CLR).

Microsoft .NET – нова платформа, яка базується на Windows. Це ціла нова парадигма програмування, яка змінить шлях, написання програм для Windows. Вона реалізована на бібліотеці класів .NET Framework class library та містить більш єдину модель програмування, покращений захист та багатші можливості для написання повнофункціональних веб-додатків.

Windows Forms – одна з найцікавіших можливостей Microsoft .NET – Windows Forms є гарним початком для роботи з .NET Framework class library, тому що вона дозволяє писати традиційні GUI додатки з вікнами, формами і т.п. речами.

Головна вигода від написання Windows-додатків з використанням Windows Forms – це те, що Windows Forms гомогенізують (створюють одноріднішу (гомогеннішу) структуру) програмну модель і усувають багато помилок і протиріч від використання Windows API. Наприклад, кожен досвідчений програміст під Windows знає, що деякі стилі вікна можуть застосовуватися лише до вікна, коли вже створено. Windows Forms значною мірою усувають таку суперечність. Якщо ви хочете існуючому вікну встановити стиль, який може бути присвоєний тільки в момент створення вікна, то Windows Forms спокійно знищить вікно і знову створить його із зазначеним стилем. Крім того, .NET Framework class library набагато

багатший, ніж Windows API, і при написанні програми, використовуючи Windows Forms, ви отримаєте в розпорядження більше можливостей. Написання програми за допомогою Windows Forms потребує менше коду, ніж програми, які використовують Windows API або MFC.

Інша вигода від Windows Forms – використання API, незалежно від мови програмування, яку ви вибрали. У минулому вибір мови програмування керував вибором API. На мові Visual Basic, викристовувався один API (реалізований мовою Visual Basic), тоді як на мові C# використовували Win32 API, а на мові C, використовувалась MFC. MFC-програмісту було важко перейти на Visual Basic і навпаки. Але тепер такого немає. Всі програми, які використовують Windows Forms, використовують один API із .NET Framework class library. Знання одного API достатньо дозволить програмісту писати програми фактично будь-якою мовою, яку він вибере.

Windows Forms не менше, ніж сучасна модель програмування для GUI додатків. На відміну від моделі програмування Win32, в якій багато що йде ще від Windows 1.0, нова модель була розроблена з урахуванням усіх сучасних вимог.

Визначення яке можна винести із вище написаного Windows Forms – інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді[11].

Всі візуальні елементи в формі форм класу Windows є похідними від класу Control. Це мінімальне функціональне значення елемента користувальницького інтерфейсу, а також загальні події, такі як миші і перетягнути / краплю. Класний контроль також повинен підтримувати док. Active Accessibility Microsoft підтримки в класі Control також допомагає знеціненими користувачам використовувати Windows, краще Forms[12].

Графічний інтерфейс користувача – це тип інтерфейсу, який дає змогу користувачам взаємодіяти з електронними пристроями через графічні

зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації, в основному виглядає як форма, що задається властивостями, які визначають їхній зовнішній вигляд, подіями які обробляють і видають реакцію на певні дії користувача які описані в методі.

Створення інтерфейсу за допомогою Windows Forms в даному дипломному проекті було виконано на мові C#, адже за допомогою стандартного набору базових елементів (TextBox «текстове поле» , «випадаючий список» ComboBox, «поле введення» LineEdit, «кнопка» Button, «радіо кнопка» RadioButton і багато інших), які можуть задовольнити практично будь які вимоги до графічного інтерфейсу значно спрощують його створення та економлять час. Також в IDE Visual Studio є необхідні обробники, та дебаггер який сильно допомагає у відладці програмного інтерфейсу, створення та прив’язка подій, зчитування та передавання даних в основні алгоритми програми, а також вивід результатів у зрозумілій формі для користувача на екран.

## 2.2 Розробка структури інтерфейсу для програмного засобу для створення індивідуальної системи тестування

При розробці інтерфейсу головної форми програмного продукту було використано кнопки що виконують функцію випадаючого меню для навігації між різними формами:

Форма авторизації має два текстові поля для введення даних користувача (логін та пароль) для входу в систему, а також кнопки для надсилання запиту на сервер та закриття форми.

Форма реєстрації має 5 текстових полів (ім’я, фамілія, е-мейл, логін, пароль) , а також кнопки для надсилання запиту на сервер та закриття форми.

Форма виведення списку усіх збережених тестів на сервері яка має 4 кнопки навігації та кнопку закриття форми.

Форма налаштування, яка відповідає за зміну мови інтерфейсу.

Форма, яка надає детальну інформацію про розробника та можливості додатку.

За допомогою властивостей форми було задано параметри, заливки форми, фону, вигляду базових елементів, додані логотипи, змінено вигляд курсору, що дало змогу зробити сучасний, актуальний дизайн програми не гірший ніж в аналогів, приємний оку користувача.

Інтерфейс програмного засобу є досить складним, так як для роботи необхідно багато елементів керування та форм. На рисунку 2.1 зображено вхідне вікно програми в Windows Forms.

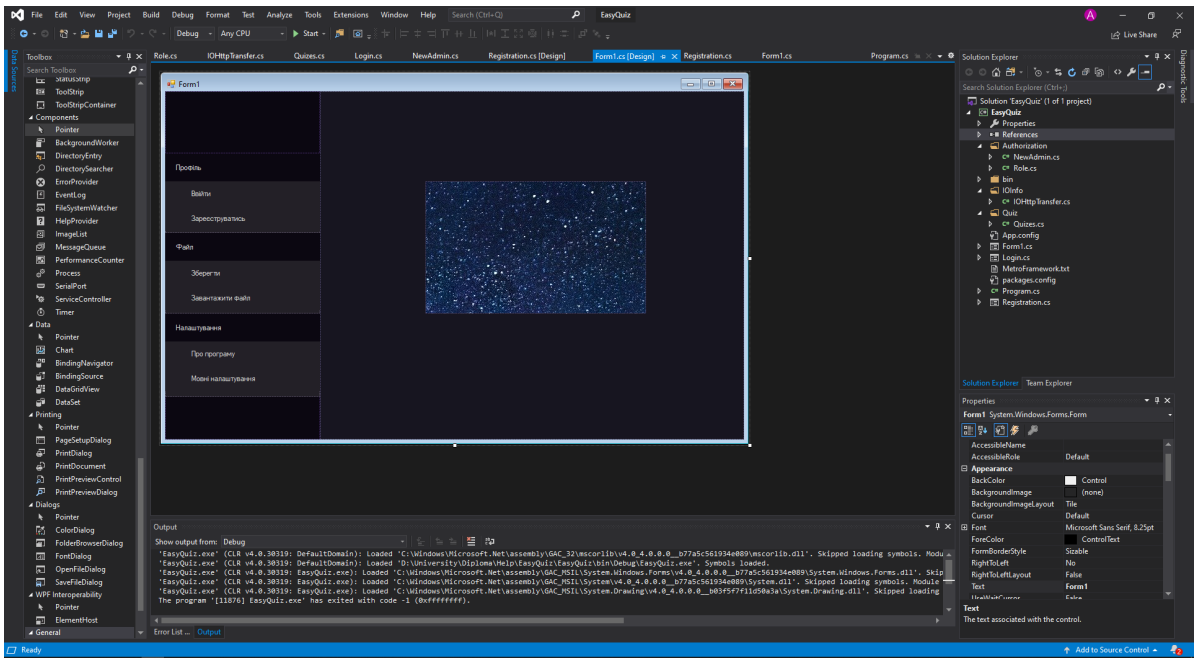


Рисунок 2.1 – Вхідне вікно програми в Windows Forms.

На рисунку 2.2 можемо побачити форму авторизації яка надає змогу надіслати запит на сервер для входження в систему та завантаження усіх існуючих тестів даного користувача.

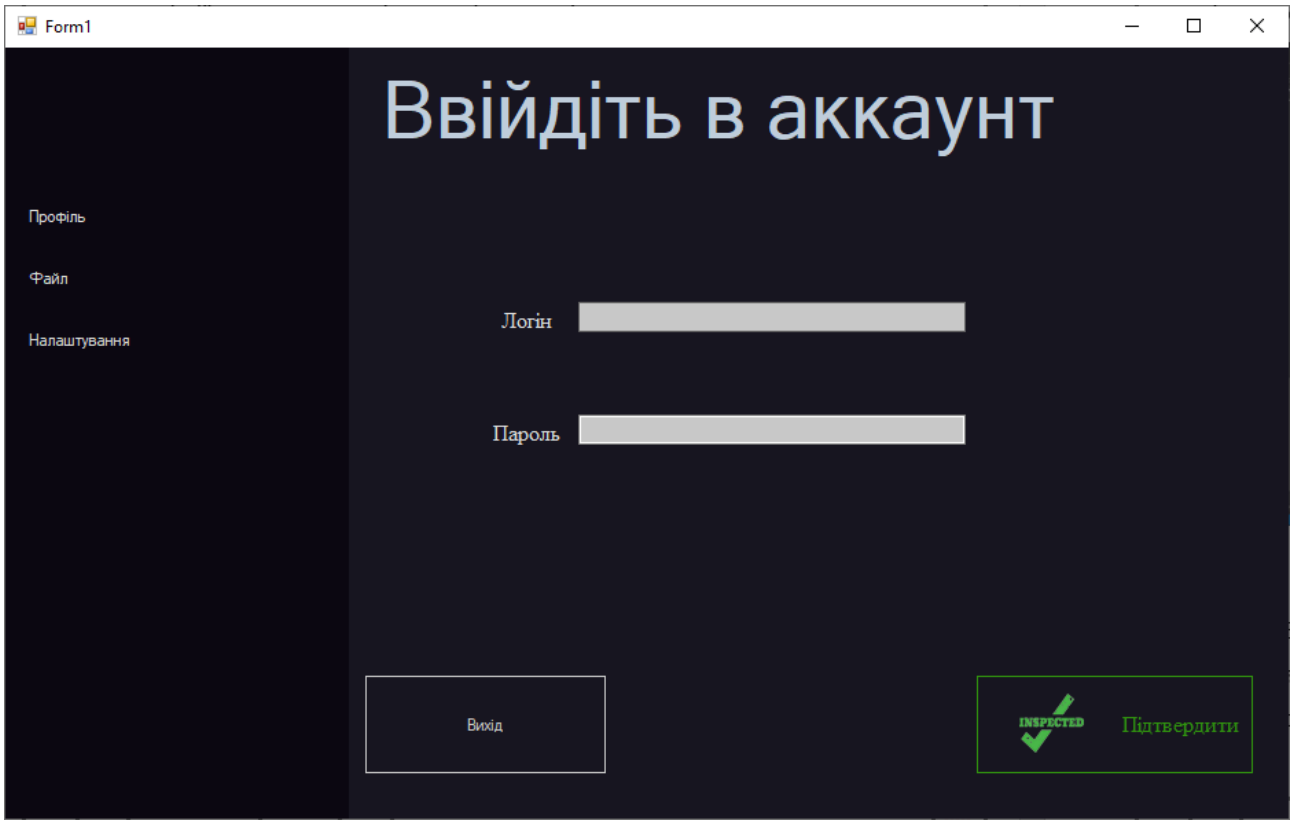



Рисунок 2.3 – Форма авторизації.

На рисунку 2.4 можемо побачити форму реєстрації яка надає змогу надіслати запит на сервер для реєстрації нового користувача в систему та можливості створення нових тестів даного користувача. Також на даній формі реалізована перевірка даних які мають співпадати з маскою створеною для них, так у полі для введення  паролю не можливо ввести буквенні дані і при їх введенні символи не будуть прийматись і навпроти поля з’явиться червоний знак, що вказуватиме користувачу на невірно введені символи, неправильно розкладку або інших помилок, які можуть бути допущені при заповненні текстових полів.

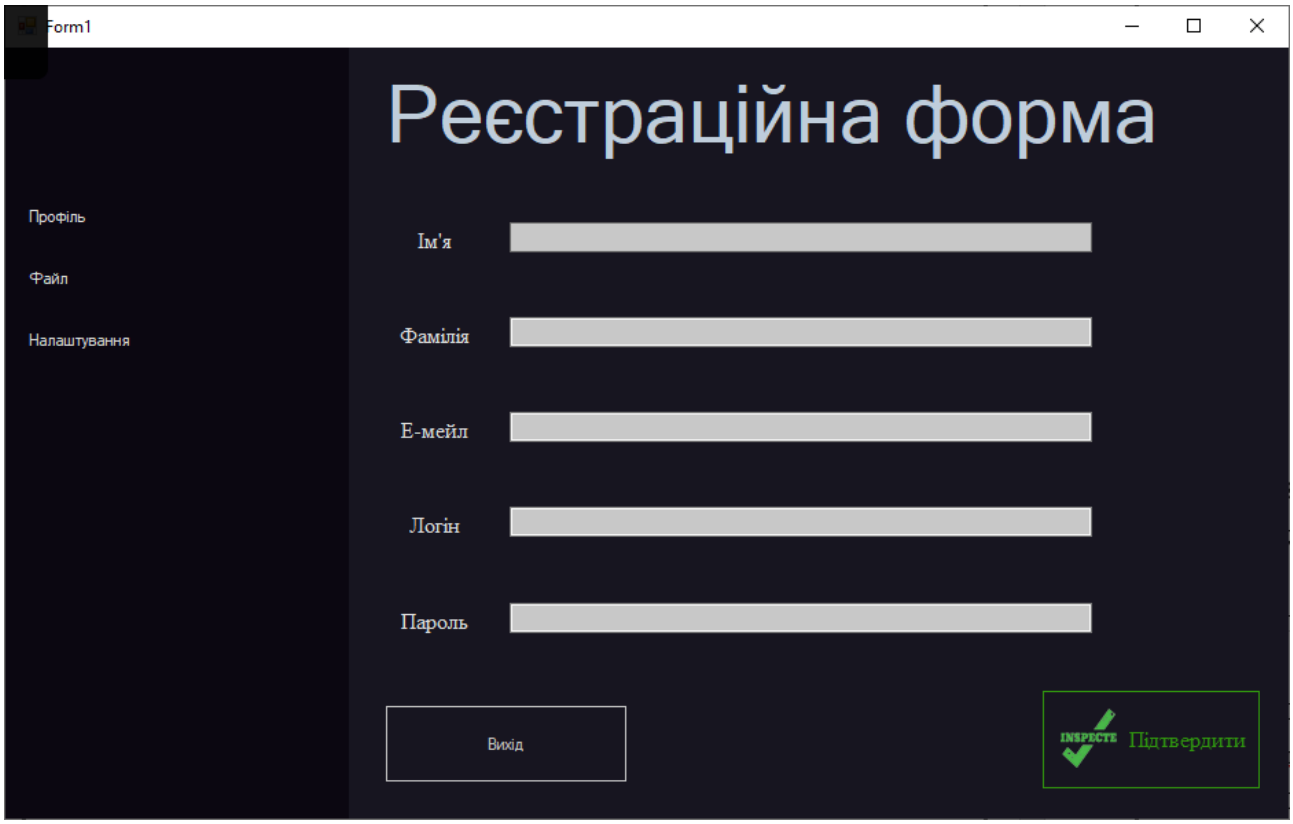



Рисунок 2.4 – Форма реєстрації.

## 2.3 Розробка алгоритмів роботи програми

Шаблон MVC описує простий спосіб побудови структури додатки, метою який допомагає відділити логіку роботи додатку від користувацького інтерфейсу. В результаті, додаток простіше масштабується, тестується, супроводжується і звичайно ж реалізується.

В архітектурі MVC модель (рис.2.5) надає дані і правила логіки, представлення відповідає за користувальницький інтерфейс, а контролер забезпечує взаємодію між моделлю і представленням.

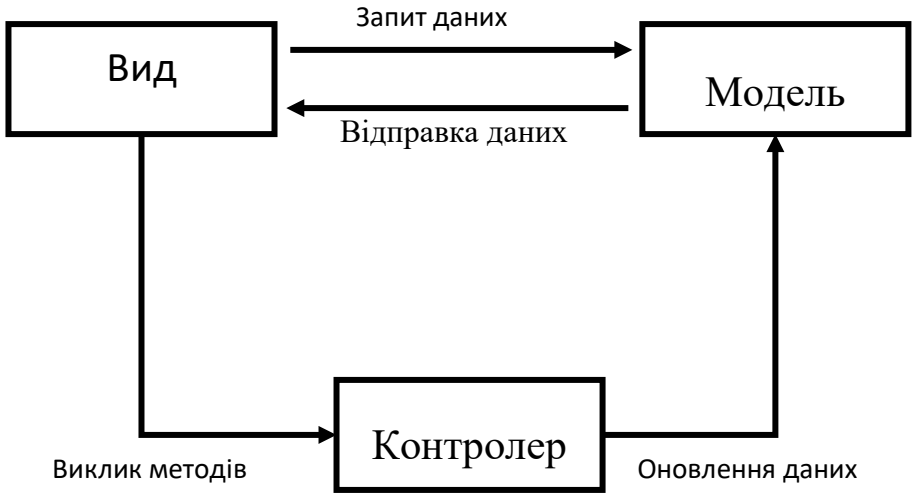



Рисунок 2.5 – Архітектура MVC

Типову послідовність роботи MVC-програми можна описати таким чином:

- Запит створюється подією що є реакцією форми Windows Forms на дії користувача.
- Додаток отримує запит від користувача і визначає запитані контролер і дію.
- Додаток створює екземпляр контролера і запускає метод дії, в якому міститися виклики моделі, що зчитують інформацію з бази даних.
- Після цього, дія формує уявлення з даними, отриманими з моделі надсилає відповідь формі на електронній машині клієнта.
- Форма приймає дані у форматі Json десеріалізує їх та виводить користувачеві в звичній текстовій формі.

Модель – містить логіку програми і включає методи вибірки (це можуть бути методи ORM), обробки (наприклад, правила перевірки) і надання конкретних даних, що часто робить її досить великою, що цілком нормально [13].

Розробка індивідуальної системи тестування полягає в тому що необхідно розробити алгоритми роботи користувацького клієнтського інтерфейсу (рисунок 2.6) та серверної частини на мові Java технології Spring (рисунок 2.7), розробити загальний алгоритм тестування (рисунок 2.8). Також

за допомогою діаграми класів можна побачити усі робочі модулі програми, взаємодію класів, описані методи і їх змінні, а також більш детально зрозуміти структуру додатку.

Наприклад, на рисунку 2.9 показано діаграму класів інтерфейсу користувача Windows Forms, а також обробку, передачу та серіалізацію даних в формат Json за допомогою бібліотеки C# в Visual Studio. Діаграму класів серверної частини додатку створеній на мові Java за допомогою Spring технології можна побачити на рисунку 2.10.

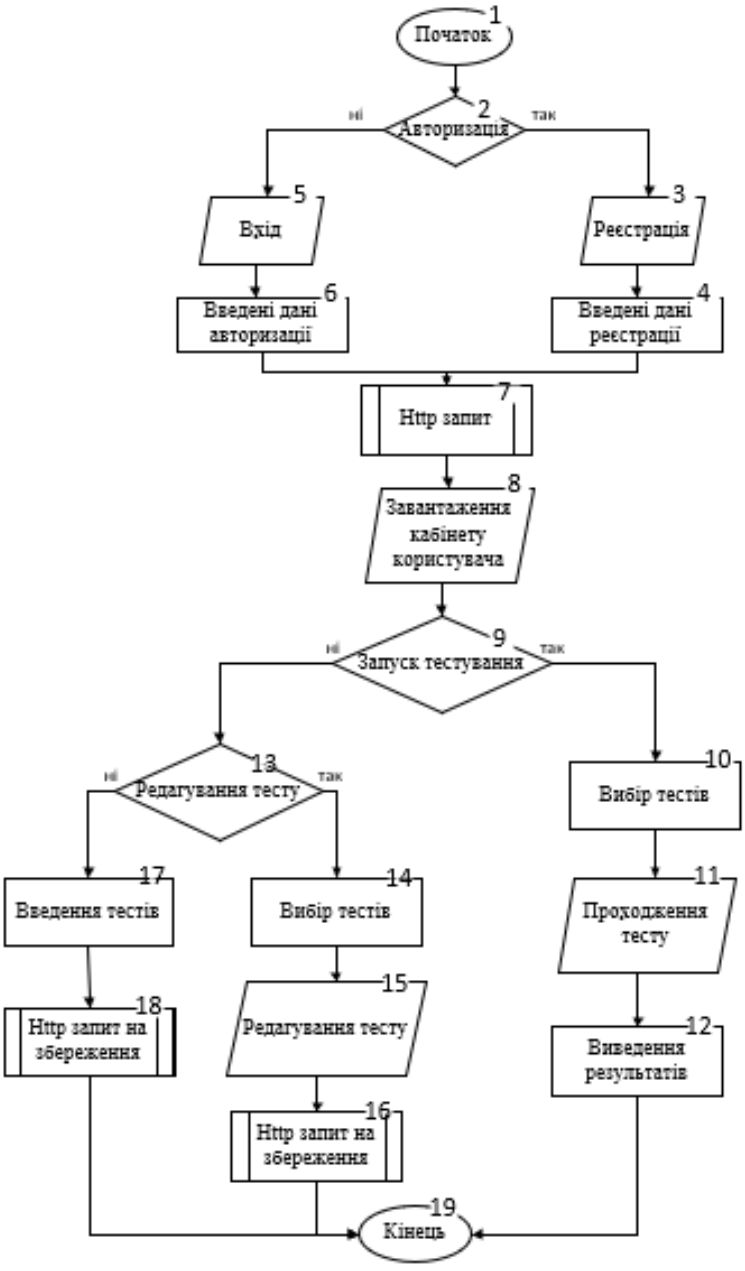



Рисунок 2.6 – Алгоритм роботи клієнтської частини додатку

Як можна зауважити з алгоритму роботи програми передача даних між двома модулями забезпечується за допомогою класичного http post запиту.

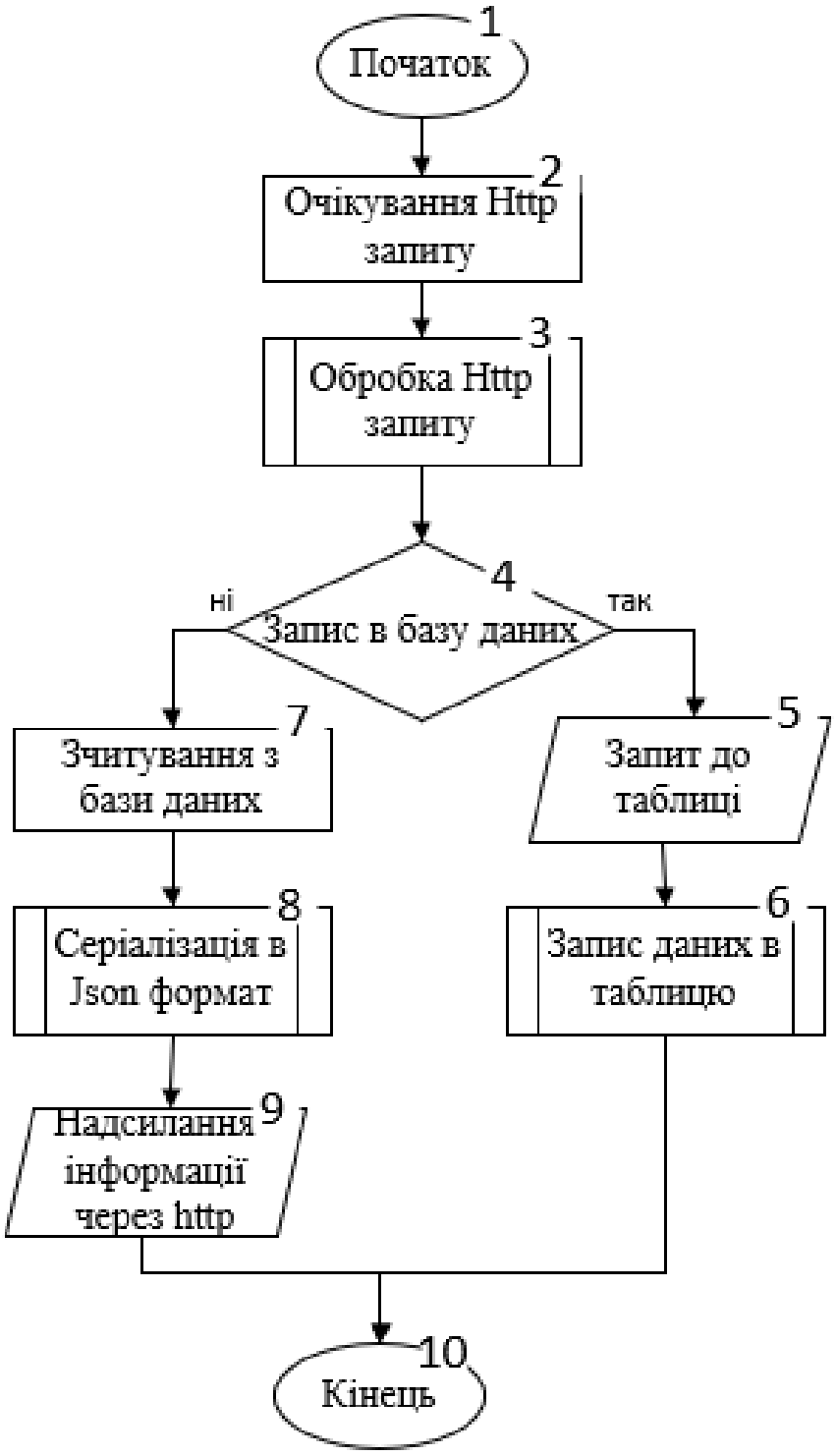



Рисунок 2.7 – Алгоритм роботи серверної частини додатку

В якості бази даних була використана вільна об’єктно-реляційна база PostgreSql.

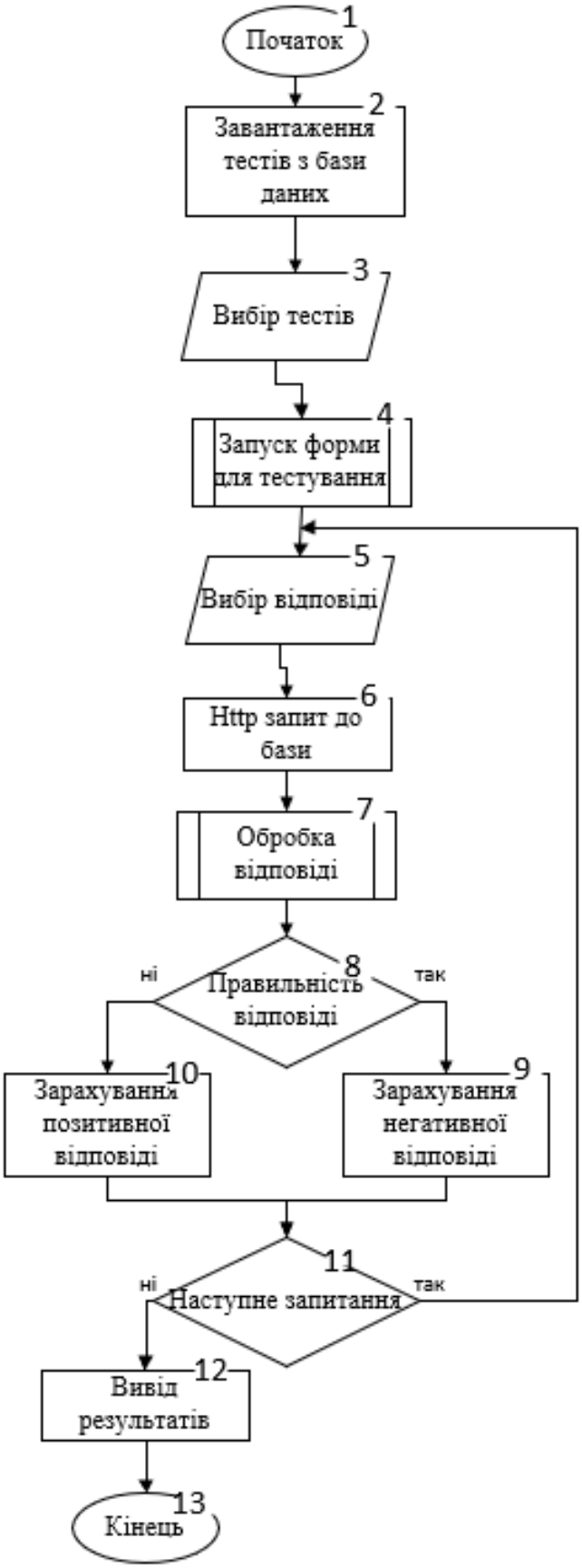



Рисунок 2.8 – Алгоритм роботи системи тестування

Діаграми класів були створенні за допомогою вбудованих можливостей IDE.

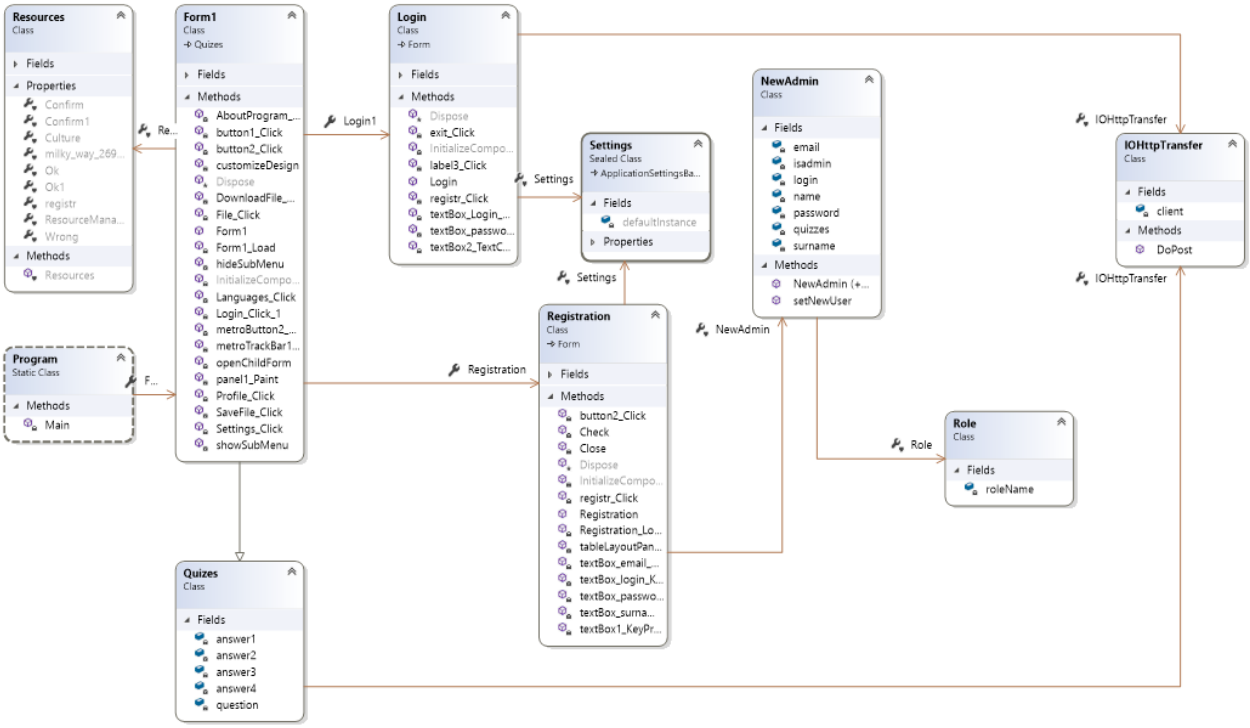



Рисунок 2.9– Діаграма класів клієнтської частини додатку Visual Studio

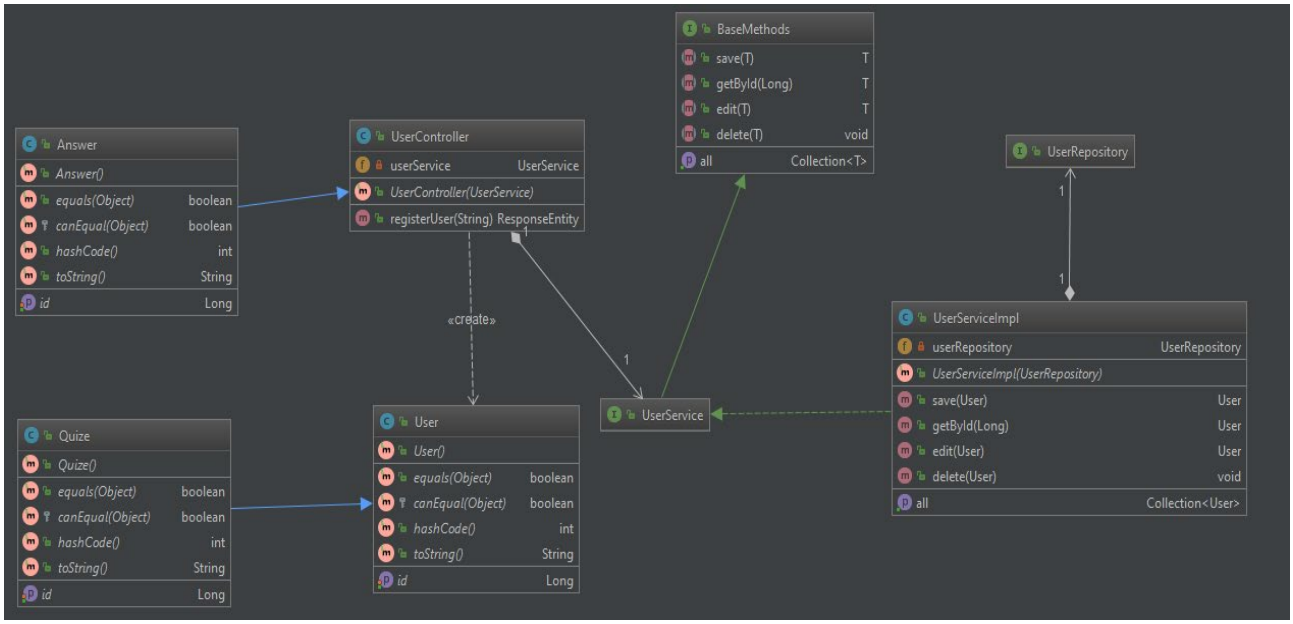



Рисунок 2.10 – Діаграма класів серверної частини додатку Intellij Idea

Детальний опис вибору даних технологій мов програмування описані в розділі «Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу».

## 2.5 Удосконалений метод автоматичного коригування складності завдань

Метод тестування із автоматичним коригуванням складності запитань, в якому складність чи інші властивості тестових завдань змінюються залежно від правильності відповідей випробуваного. Наприклад, якщо студент правильно відповідає на тестові завдання, складність наступних завдань підвищується, якщо неправильно – знижується. Також є можливість внесення додаткових питань в області, яку опитуваний погано знає для більш тонкого з'ясування рівня знань у цих галузях. Системи адаптивного тестування визнаються більш ефективними за існуючі класичні системи, так як вручну важко знайти ту відповідність знанням та визначити порядок видачі запитань в залежності від рівня знань опитуваного.

Відомі наступні варіанти тестування.

– Всім користувачам на початку тестування даються завдання середньої складності й потім, у залежності від відповідей, опитуваному дається завдання складніше чи простіше.

– Контроль знань починається з будь-якого рівня складності завдань і далі ітераційно наближається до реального рівня складності, що відповідає рівню знань.

– Існує база знань, з питаннями, розподіленими по рівнях складності. При правильній відповіді наступне завдання обирається із групи більш складних завдань, при неправильному – з більш легких.

– Тестування починається з перевірки знання найважливішого терміну та у випадку коректних відповідей підвищується складність шляхом перевірки знання менш важливих термінів.

– Обхід за структурою НК зверху вниз, у випадку невірної відповіді на найбільш прості питання тестування у частині структури, що слідує нижче, не проводиться..

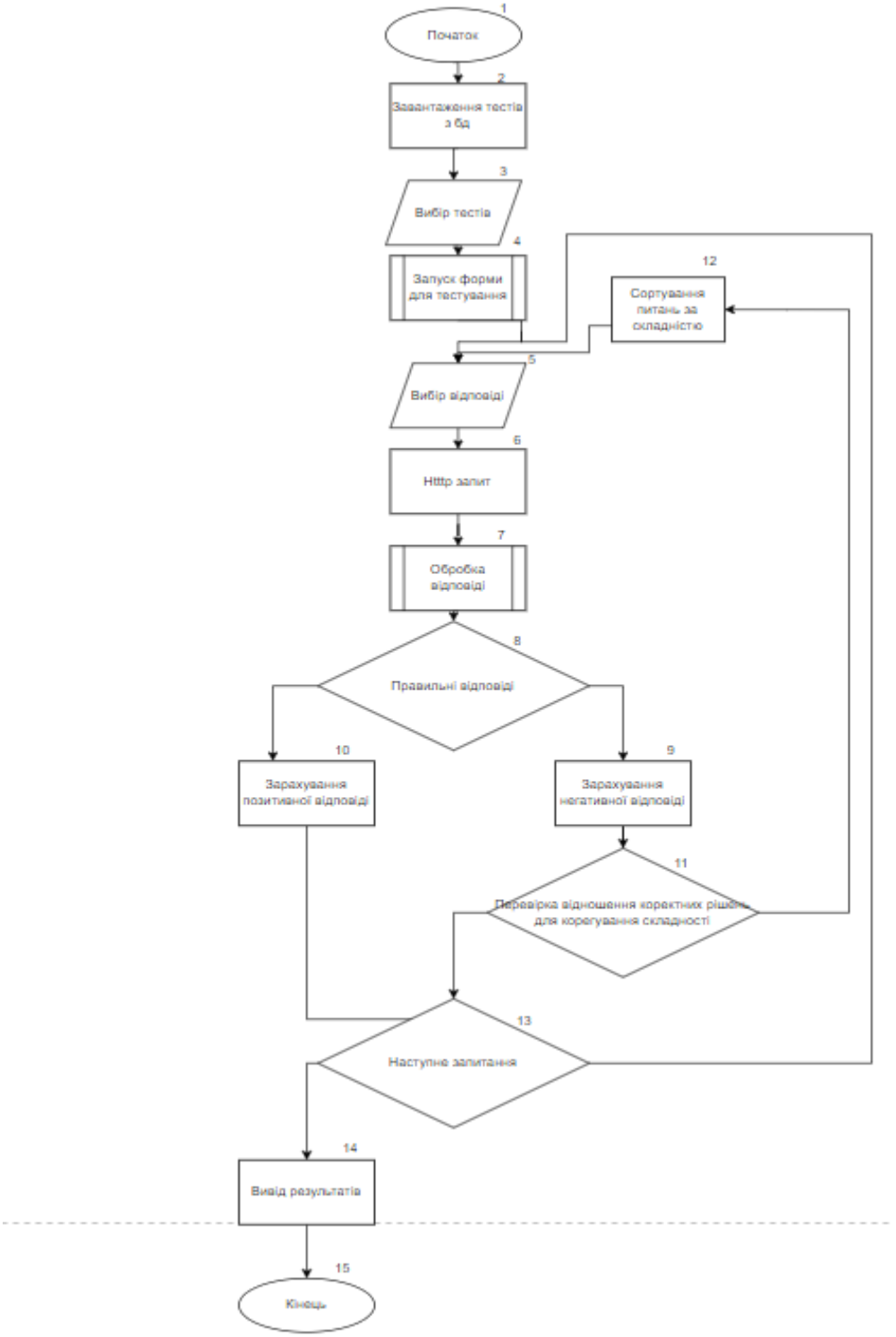



Рисунок 2.11 – Алгоритм роботи системи коригування рівня складності запитань

Наведений алгоритм гнучкого вибору тестових запитань (рис. 2.11) дозволяє проводити тестування за максимальною кількістю елементів структури навчального матеріалу, розпочинаючи тестування від найбільш загального терміну.

Складність запитань зростає у випадку коректної відповіді. Якщо ж користувач не спроможний дати вірну відповідь навіть на найлегше запитання, то немає сенсу продовжувати опитування блоку та перевіряти знання на складніших питаннях. Тому повна перевірка знань елементів структури навчального матеріалу закінчується достроково у разі некоректних відповідей або ж закінчується після коректних відповідей на усі запитання, включаючи найскладніші.

## 2.6 Висновки

У другому розділі було показано приклад користувацького інтерфейсу додатку EasyQuiz, а також технології за допомогою якої було його створено, виконані всі вимоги щодо зручності, сучасного вигляду та актуальності даного інтерфейсу в Windows Forms. Розроблено діаграми алгоритмів роботи програми, показана діаграма класів двох основних компонентів програми (клієнтської і серверної), за допомогою якої можна побачити саму структуру програми, взаємодію модулів, методів та змінних. Описано удосконалений метод автоматичного коригування складності завдань, який дає можливість

# 3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ ІНДИВІДУАЛЬНОЇ СИСТЕМИ ТЕСТУВАННЯ.

## 3.1 Варіантний аналіз та обґрунтування засобів реалізації

Для розробки будь-якого програмного продукту головним є вибір мови програмування та технології яка буде використана при розробці, від цього залежить наскільки добре будуть виконані основні вимоги до додатку і наскільки та чи інша технологія мови задовольняє потреби розробника. Для реалізації даного програмного продукту було використано дві мови програмуванні відповідно зі своїми технологіями. Для розробки інтерфейсу було використано технологію Windows Forms на мові C#, а також Java з технологією Spring. Так як в IDE Visual Studio дуже швидко можна створити інтерфейс програми, налаштувати його візуальний вигляд, всі базові елементи керування присутні та їх легко можна прив’язати до подій, що одразу дає змогу створити робочий інтерфейс що буде реагувати на дії користувача. Така технологія як Spring дає змогу легко підняти сервер отримувати запити, має всі необхідні інструменти Java для роботи з базою даних та швидкого її налаштування. Порівняльні характеристики наведені у таблиці 3.1.

Ruby

Розробник – Юкіхіро Нацумото

Остання версія – 2.5.1

Рік появи - 1993

Ліцензія – BSD

Ruby – це інтерпретована, повністю об'єктно-орієнтована мова програмування з чіткою динамічною типізацією. Мова вирізняється високою ефективністю розробки програм і увібрала в себе найкращі риси Perl, Java, Python, Smalltalk, Eiffel, Ada і Lisp. Ruby поєднує в собі Perl-подібний синтаксис із об'єктно-орієнтованим підходом мови

програмування Smalltalk. Також деякі риси запозичено із мов програмування Python, Lisp, Dylan та CLU [14].

Java

Розробник – Sun Microsystems, Oracle

Остання версія – Java Standart Edition 10

Рік появи - 1995

Ліцензія – GNU General Public License

Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування[15].

C++

Розробник – Б’ярн Страуструп

Остання версія - ISO/IEC 14882:2017

Рік появи – 1985

C++ (Сі-плюс-плюс) – мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Згодом Страуструп перейменував мову на C++ у 1983 р. Базується на мові C. У 1990-х роках C++ стала однією з найуживаніших

мов програмування загального призначення. Мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм, наприклад, відеоігор. C++ суттєво вплинула на інші популярні сьогодні мови програмування: C# та Java[16].

Результати порівняння мов програмування зведені в табл. 3.1.

Таблиця 3.1 – Порівняння мов програмування

| Властивості | Мови | | | | |
|---|---|---|---|---|---|
| | C++ | Basic | Pascal | Java | C# |
| Імперативність | + | + | + | + | + |
| Об’єктно-орієнтованість | + | - | + | + | + |
| Функціональність | +/- | + | +/- | + | + |
| Узагальненість програмування | + | - | - | + | + |
| Розподіленість | + | - | - | + | + |

Імперативне програмування – парадигма програмування, згідно з якою описується процес отримання результатів як послідовність інструкцій зміни стану програми. Подібно до того, як з допомогою наказового способу в мовознавстві перелічується послідовність дій, що необхідно виконати, імперативні програми є послідовністю операцій комп'ютеру для виконання. Поширений синонім імперативному програмуванню є процедурне програмування [17].

Об'єктно – орієнтоване програмування (ООП) – це модель програмування яка базується на стверджені того, що програма це сукупність об’єктів які взаємодіють між собою. Кожен об’єкт в цій моделі є незалежним, і він здатний отримувати, обробляти дані та відправляти ці дані іншим

об’єктам. В ООП використано моделі успадкування, модульності, поліморфізму та інкапсуляції [18].

Наслідування є однією з фундаментальних основ об’єктно-орієнтованого програмування, у якому клас “отримує” всі атрибути і операції класу, нащадком якого він є, і може також додавати власні атрибути і операції [19].

Функціональне програмування - це програмування, в якому функції є об'єктами, і їх можна присвоювати змінним, передавати в якості аргументів інших функцій, повертати як результат від функцій і т. п.

Узагальнене програмування – парадигма програмування, що полягає в такому описі даних і алгоритмів, який можна застосовувати до різних типів даних, не змінюючи самий цей опис.

Розподілене програмування в малому пов'язане з паралельним виконанням операторів однієї програми (наприклад, паралельним виконанням циклів). Технологія розробки розподілених програм пов'язана з реалізацією комунікаційних пакетів, а також мов програмування і бібліотек, що полегшують використання таких пакетів.

Серіалізація[20] – процес перетворення будь-якої структури даних в послідовність бітів. Зворотною до операції серіалізації є операція десеріалізації - відновлення початкового стану структури даних з бітової послідовності.

Серіалізація надає декілька корисних можливостей:

- метод реалізації зберігання об'єктів, який зручніший, ніж запис їх властивостей в текстовий файл на диск і повторна збірка об'єктів читанням файлів;
- метод здійснення віддалених викликів процедур, як, наприклад, у SOAP;
- метод розповсюдження об'єктів, особливо в технологіях компонентно-орієнтованого програмування, таких як COM і CORBA;
- метод виявлення змін у даних, що змінюються з часом.

Для найефективнішого використання даних можливостей необхідно підтримувати незалежність від архітектури. Наприклад, необхідно мати можливість надійно відтворювати серіалізований потік даних, незалежно від порядку байтів, що використовується в цій архітектурі. Це означає, що найбільш проста і швидка процедура прямого копіювання ділянки пам'яті, в якому розміщується структура даних, не може працювати надійно для всіх архітектур. Серіалізація структур даних в архітектурно-незалежний формат означає, що не повинно виникати проблем через різний порядок проходження байтів, механізмів розподілу пам'яті або відмінностей представлення структур даних в мовах програмування [21].

Будь-якій зі схем серіалізації властиво те, що кодування даних послідовно за визначенням, і вибірка будь-якої частини серіалізованої структури даних вимагає, щоб весь об'єкт був зчитаний від початку до кінця і був відновлений. У багатьох програмах така лінійність корисна, тому що дозволяє використовувати прості інтерфейси введення/виведення загального призначення для збереження і передачі стану об'єкта. У додатках, де важлива висока продуктивність, можливо буде доречніше використовувати складнішу, нелінійну організацію зберігання даних.

Spring Framework – це програмний каркас з відкритим кодом та контейнери з підтримкою для платформи Java.

Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але є розширення для створення веб-додатків на платформі Java EE. Незважаючи на це, Spring Framework не нав'язує якоїсь конкретної моделі програмування, Spring Framework став популярним в спільноті Java як альтернатива, або навіть доповнення моделі [22].

Windows Forms – це інтелектуальна клієнтська технологія для .NET Framework, набір керованих бібліотек, що спрощують типові завдання програми, такі як читання і запис в файлову систему. При використанні середовища розробки, такий як Visual Studio, можна створювати Windows Forms інтелектуальні клієнтські програми, які відображають відомості,

запитують введення даних від користувачів і обмінюються даними з віддаленими комп'ютерами по мережі.

У Windows Forms форма – це візуальна поверхня, на якій виводиться інформація для користувача. Зазвичай додаток Windows Forms будується шляхом приміщення елементів управління на форму і написання коду для реагування на дії користувача, такі як клацання миші або натискання клавіш. Елемент управління – це окремий елемент призначеного для користувача інтерфейсу, призначений для відображення або введення даних.

При виконанні користувачем якої-небудь дії з формою або одним з її елементів управління створюється подія. Додаток реагує на ці події за допомогою коду і обробляє події при їх виникненні.

За допомогою конструктору Windows Forms в Visual Studio можна легко створювати Windows Forms додатки. Досить виділити елемент керування курсором і помістити його в потрібне місце на формі. Для подолання труднощів, пов'язаних з вирівнюванням елементів управління, конструктор надає такі інструменти, як лінії сітки і лінії прив'язки. А також при використанні Visual Studio або компіляції з командного рядка можна використовувати елементи управління FlowLayoutPanel, TableLayoutPanel і SplitContainer для створення розширених макетів форм за менший час [23].

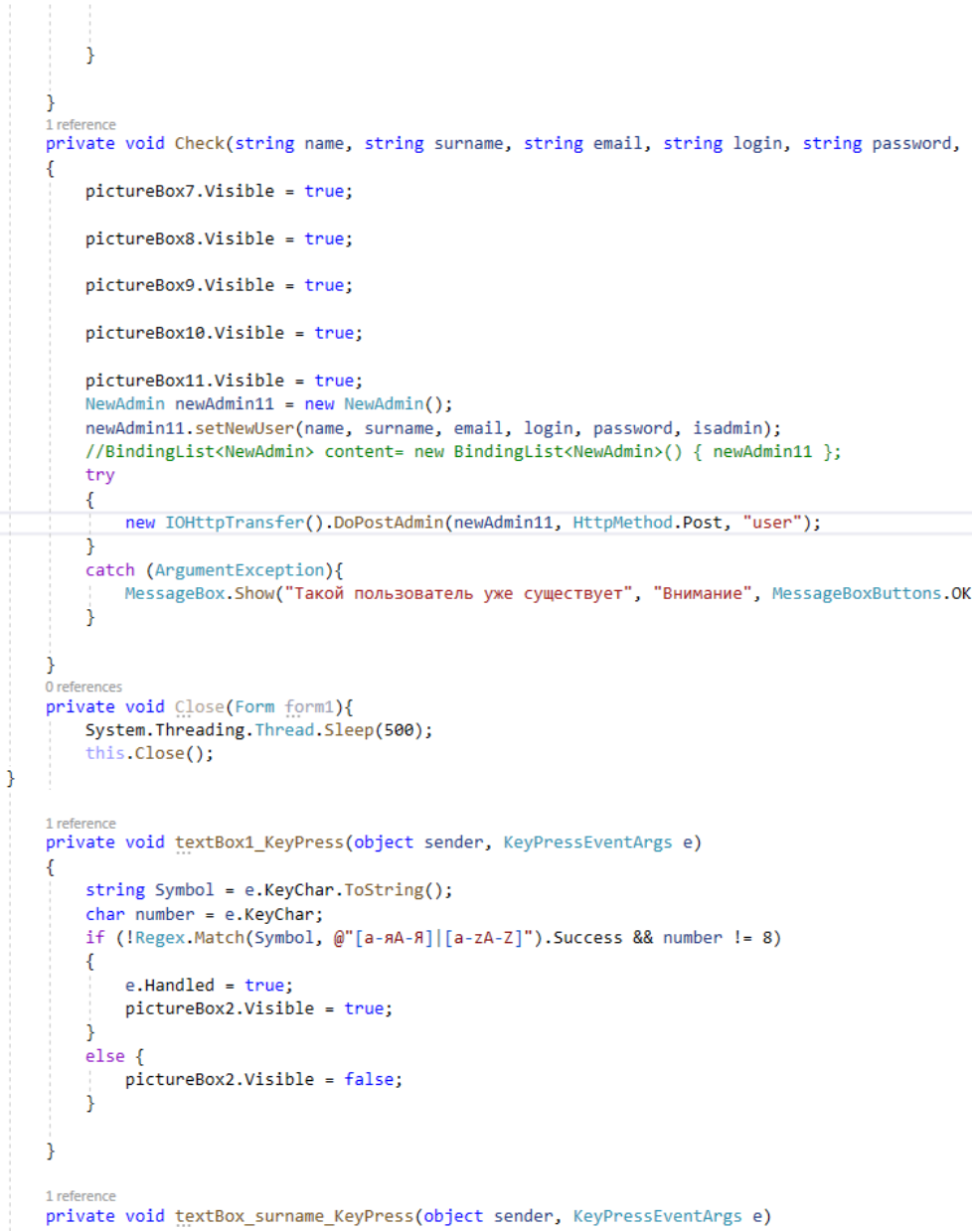
Враховуючи всі вищенаведені переваги, можна сказати, що мови програмування Java та C# є найбільш доцільними для розробки програмного засобу.

## 3.2 Розробка авторизаційного блоку користувацького інтерфейсу

Авторизація – керування рівнями та засобами доступу до певного захищеного ресурсу, як у фізичному розумінні (доступ до кімнати готелю за карткою), так і в галузі цифрових технологій (наприклад, автоматизована система контролю доступу) та ресурсів системи залежно від ідентифікатора і

пароля користувача або надання певних повноважень (особі, програмі) на виконання деяких дій у системі обробки даних [24].

Реєстрація нового користувача починається із заповнення полів в реєстраційній формі. Розглянемо код головної форми реєстрації на рисунку 3.1



```
        }

    }
    1 reference
    private void Check(string name, string surname, string email, string login, string password,
    {
        pictureBox7.Visible = true;

        pictureBox8.Visible = true;

        pictureBox9.Visible = true;

        pictureBox10.Visible = true;

        pictureBox11.Visible = true;
        NewAdmin newAdmin11 = new NewAdmin();
        newAdmin11.setNewUser(name, surname, email, login, password, isadmin);
        //BindingList<NewAdmin> content= new BindingList<NewAdmin>() { newAdmin11 };
        try
        {
            new IOHttpTransfer().DoPostAdmin(newAdmin11, HttpMethod.Post, "user");
        }
        catch (ArgumentException){
            MessageBox.Show("Такой пользователь уже существует", "Внимание", MessageBoxButtons.OK
        }

    }
    0 references
    private void Close(Form form1){
        System.Threading.Thread.Sleep(500);
        this.Close();
}

    1 reference
    private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
    {
        string Symbol = e.KeyChar.ToString();
        char number = e.KeyChar;
        if (!Regex.Match(Symbol, @"[а-яА-Я]|[a-zA-Z]").Success && number != 8)
        {
            e.Handled = true;
            pictureBox2.Visible = true;
        }
        else {
            pictureBox2.Visible = false;
        }

    }

    1 reference
    private void textBox_surname_KeyPress(object sender, KeyPressEventArgs e)
```

Рисунок 3.1 – Лістинг коду форми реєстрації

Для створення об’єкту класу NewAdmin необхідно заповнити такі дані для реєстрації:

– Ім’я користувача
– Прізвище користувача

– Е-мейл користувача
– Логін користувача
– Пароль користувача

Розглянемо Клас користувача на рисунку 3.2

```
[DataContract]
[Serializable]
9 references
class NewAdmin
{
    [DataMember]
    string name;
    [DataMember]
    string surname;
    [DataMember]
    string email;
    [DataMember]
    string username;
    [DataMember]
    string password;
    [DataMember]
    List<Quizes> quizzes;
    [DataMember]
    bool isadmin;

    2 references
    public string Name { get => name; set => name = value; }
    2 references
    public string Surname { get => surname; set => surname = value; }
    2 references
    public string Email { get => email; set => email = value; }
    4 references
    public string Username { get => username; set => username = value; }
    3 references
    public string Password { get => password; set => password = value; }
    1 reference
    internal List<Quizes> Quizzes { get => quizzes; set => quizzes = value; }
    2 references
    public bool Isadmin { get => isadmin; set => isadmin = value; }

    2 references
    public NewAdmin()
    {
    }

    0 references
    public NewAdmin(string name, string surname, string email, string login, string password, List<Quizes> quizzes, bool isadmin)
    {
        this.Name = name;
        this.Surname = surname;
        this.Email = email;
        this.Username = login;
        this.Password = password;
        this.Quizzes = quizzes;
        this.Isadmin = isadmin;
    }

    1 reference
    public void setNewUser(string name, string surname, string email, string login, string password, bool isadmin)
    {
        this.Name = name;
        this.Surname = surname;
        this.Email = email;
        this.Username = login;
        this.Password = password;
        this.Isadmin = isadmin;
        Console.WriteLine(name, surname, email, login, password, isadmin);
```

Рисунок 3.2 – Лістинг коду класу користувача

Наступний кроком є створення http запиту та серіалізації об’єкту в формат Json. Розглянемо код передачі даних за допомогою http запиту на рисунку 3.3.

```
class IOHttpTransfer
{
    public bool isexist;
    private static readonly HttpClient client = new HttpClient();


    2 references
    public void DoPostAdmin(NewAdmin jsonContent, HttpMethod method, string urlPath) {
        DataContractJsonSerializer formatter = new DataContractJsonSerializer(typeof(NewAdmin));

        string serialized = JsonConvert.SerializeObject(jsonContent);
        var json = JsonConvert.SerializeObject(jsonContent);

        HttpClient client = new HttpClient();
        client.BaseAddress = new Uri("http://localhost:8080");

        client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));//ACCEPT header

        HttpRequestMessage request = new HttpRequestMessage(method, urlPath);
        request.Content = new StringContent(json, Encoding.UTF8, "application/json");
        client.SendAsync(request).ContinueWith(responseTask =>
        {
            if (!responseTask.Result.IsSuccessStatusCode) {
                throw new ArgumentException();

            };


        });
    }
    1 reference
    public String DoGetIdAdmin(HttpMethod method, string urlPath)
    {
        DataContractJsonSerializer formatter = new DataContractJsonSerializer(typeof(NewAdmin));

        WebRequest request = WebRequest.Create(urlPath);

        using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
        using (Stream stream = response.GetResponseStream())
        using (StreamReader reader = new StreamReader(stream))
        {
            return reader.ReadToEnd();
        }


    }
    1 reference
    public void DoPostQuizes(Quizes jsonContent, HttpMethod method, string urlPath)
    {
        DataContractJsonSerializer formatter = new DataContractJsonSerializer(typeof(Quizes));

        string serialized = JsonConvert.SerializeObject(jsonContent);
        var json = JsonConvert.SerializeObject(jsonContent);

        HttpClient client = new HttpClient();
        client.BaseAddress = new Uri("http://localhost:8080");

        client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));//ACCEPT header

        HttpRequestMessage request = new HttpRequestMessage(method, urlPath);
        request.Content = new StringContent(json, Encoding.UTF8, "application/json");
        client.SendAsync(request).ContinueWith(responseTask =>
        {
            if (!responseTask.Result.IsSuccessStatusCode)
            {
                throw new ArgumentException();
            };

        });
```

Рисунок 3.3 – Лістинг блоку коду, що відповідає за HTTP запити.

JSON (JavaScript Object Notation) – це загальний формат для представлення значень і об'єктів. Його опис задокументовано в стандарті RFC

4627. Спочатку він був створений для JavaScript, але багато інших мов також мають бібліотеки, які можуть працювати з ним. Таким чином, JSON легко використовувати для обміну даними, коли клієнт використовує JavaScript, а сервер написаний на Ruby / PHP / Java або будь-якому іншому мовою[25].

Він читабельність і економніше, ніж xml. Різноманіття бібліотек для роботи з json на сьогоднішній день майже не поступається бібліотекам для роботи з xml. Для Java – org.json.JSONObject або GSON, для C # – Json.NET, DataContractJsonSerializer [26]. Допустимі типи в запису об'єкта json:

- число
- рядок
- літерали null, false, true
- об'єкт json (вкладене визначення)
- одновимірний масив (об'єктів (число, рядок, літерал, об'єкт json)).

## 3.3 Розробка серверної частини програмного додатку для проведення авторизації

Серверна частина програми була написана за допомогою технології Spring на мові Java, а також в якості бази даних було використано вільну об’єктно-реляційну базу PostgreSQL.

PostgreSQL – широко розповсюджена система керування базами даних з відкритим сирцевим кодом. Прототип був розроблений в Каліфорнійському університеті Берклі в 1987 році під назвою POSTGRES, після чого активно розвивався і доповнювався. В червні 1990 року з'явилась друга версія із переробленою системою правил маніпулювання та роботи з таблицями, у 1991 році – третя версія, із доданою підтримкою одночасної роботи кількох менеджерів збереження, покращеним механізмом запитів і доповненою системою внутрішніх правил. В цей час POSTGRES використовувався для реалізації великих систем, таких як: система аналізу фінансових даних, пакет моніторингу функціональності потоків, база даних відстеження астероїдів,

система медичної інформації, кілька географічних систем. POSTGRES також використовувався як навчальний інструмент в кількох університетах. 1992 року POSTGRES став головною СКБД наукового комп'ютерного проекту Sequoia 2000. 1993 року кількість користувачів подвоїлась. Стало зрозуміло, що для підтримки й подальшого розвитку необхідні великі витрати часу на дослідження баз даних, тому офіційно проект Берклі було зупинено на версії 4.2. 1994 року Andrew Yu і Jolly Chen додали інтерпретатор мови SQL, вдосконалили сирцевий код і виклали в Інтернеті свою реалізацію під назвою Postgres95. 1996 року програмний продукт було перейменовано на PostgreSQL із початковою версією 6.0. Подальшою підтримкою й розробкою займається група спеціалістів у галузі баз даних, які добровільно приєднались до цього проекту[27].

Також було використано асинхронні запити для обміном інформацією з базою даних, але спочатку потрібно розібратись що таке HTTP запити

Даний протокол визначає взаємодію між двома комп'ютерами (клієнтом і сервером), побудоване з урахуванням повідомлень, званих запит (Request) і відповідь (Response). Кожне повідомлення складається з трьох частин: стартовий рядок, заголовки та тіло. При цьому обов'язковим є лише стартовий рядок.

Стартові рядки для запиту та відповіді мають різний формат – нас цікавий тільки стартовий рядок запиту, який виглядає так:

METHOD URI HTTP/VERSION, де METHOD – це якраз метод HTTP-запиту, URI – ідентифікатор ресурсу, VERSION – версія протоколу (на даний момент актуальна версія 1.1).

Заголовки – це набір пар ім'я-значення, розділених двокрапкою. У заголовках передається різна службова інформація: кодування повідомлення, назва та версія браузера, адреса, з якої прийшов клієнт (Referrer) тощо.

Тіло повідомлення – це, власне, дані, що передаються. У відповіді переданими даними, зазвичай, є html-сторінка, яку запросив браузер, а запиті, наприклад, у тілі повідомлення передається вміст файлів, завантажуваних на

сервер. Але, як правило, тіло повідомлення у запиті взагалі відсутнє. Uniform Resource Identifier – одноманітний ідентифікатор ресурсу. Ресурс – це, як правило, файл на сервері (приклад URI в даному випадку '/styles.css'), але взагалі ресурсом може бути якийсь абстрактний об'єкт ('/blogs/webdev/' – вказує на блок «Веб- технологія», а чи не на конкретний файл).

Тип HTTP-запиту (також званий HTTP-метод) вказує серверу те, яке дію хочемо зробити з ресурсом. Спочатку (на початку 90-х) передбачалося, що клієнт може хотіти від ресурсу лише одне – отримати його, проте зараз за протоколом HTTP можна створювати пости, редагувати профіль, видаляти повідомлення та багато іншого. І ці дії складно поєднати терміном «отримання».

Для розмежування дій із ресурсами лише на рівні HTTP-методів і було винайдено такі варианты:

- GET отримання ресурсу
- POST створення ресурсу
- PUT оновлення ресурсу
- DELETE видалення ресурсу

HTTP метод GET використовується для отримання (або читання) представлення ресурсу. У разі вдалої (або не містить помилок) адреси GET повертається представлення ресурсу у форматі XML або JSON у поєднанні з кодом стану HTTP 200 (OK). У разі помилок зазвичай повертається код 404 (NOT FOUND) або 400 (BAD REQUEST).

Відповідно до специфікації HTTP, GET (як і HEAD) запити використовуються лише читання даних, не змінюючи їх. Таким чином, при дотриманні цієї угоди вони вважаються безпечними. Тобто вони можуть використовуватися без ризику зміни даних, незалежно від того, один раз дані були отримані, або 10, або жодного разу зовсім. GET (а також HEAD) запити є ідемпотентними (тотожними), що передбачає отримання ідентичних даних при використанні тих самих запитів (як при одиничному зверненні, так і при багаторазовому).

Не варто використовувати GET для небезпечних операцій над даними, при даному запиті вони не повинні бути змінені.

Метод PUT зазвичай використовується для надання можливості оновлення ресурсу. Тіло запиту при надсиланні PUT-запиту до існуючого ресурсу URI має містити оновлені дані оригінального ресурсу (повністю або лише оновлювану частину).

Крім того, PUT може бути використаний для створення ресурсу, якщо ідентифікатор ресурсу вибирає клієнт, а не сервер. Або, якщо перефразувати — при надсиланні PUT запиту на адресу, що містить ідентифікатор ресурсу, що не існує. Знову ж таки, варто пам'ятати, що тіло запиту має бути модифікацією оригінального ресурсу. Багато хто вважає це заплутаним та незрозумілим. Відповідно, цю можливість методу PUT варто використовувати з обережністю. Та й за крайньої необхідності.

Для створення нових екземплярів ресурсу краще використовувати POST запиту. В даному випадку, при створенні екземпляра буде надано коректний ідентифікатор екземпляра ресурсу повернутих даних про екземпляра.

При успішному оновленні за допомогою виконання PUT запиту повертається код 200 (або 204 якщо не було передано будь-який контент у тілі відповіді). Якщо PUT використовується для створення екземпляра, зазвичай повертають HTTP код 201 при успішному створенні. Повертати дані у відповідь на запит необов'язково. Також не обов'язково повертати посилання на екземпляр ресурсу за допомогою заголовка Location через те, що клієнт і так має ідентифікатор екземпляра ресурсу.

PUT не є безпечною операцією, оскільки внаслідок її виконання відбувається модифікація (або створення) екземплярів ресурсу на стороні сервера, але цей метод ідемпотентний. Іншими словами, створення або оновлення ресурсу за допомогою відправки PUT запиту - ресурс не зникне, буде розташовуватися там же, де і був при першому зверненні, а також багаторазове виконання одного і того ж PUT запиту не змінить загального

стану системи (за винятком першого разу, але це зазвичай опускають із розгляду)

Якщо PUT запит використовується для збільшення лічильника перегляду конкретного ресурсу, цей запит вже не вважається ідемпотентним. Іноді таке відбувається і вважається достатнім задокументувати те, що виклик не ідемпотентний. Проте суворо рекомендується витримувати ідемпотентність запиту PUT.

PATCH запит використовується для модифікації ресурсу. PATCH запит повинен містити лише дані ресурсу, що змінюються, а не всі його дані.

Це нагадує роботу PUT запиту, але в тілі запиту міститься набір інструкцій, що описують як повинен бути змінений ресурс, розташований на сервері, для формування нової версії. Це означає, що тіло PATCH запиту має містити не просто зміни ресурсу, а являти собою опис мовою внесення змін (patch language), таких як JSON Patch або XML Patch.

PATCH запит не є безпечним, ні ідемпотентним. Однак PATCH запит може бути сформований таким чином щоб бути ідемпотентним, що в свою чергу допомагає запобігти негативним наслідкам від колізій між двома PATCH запитами до одного і того ж ресурсу в той самий проміжок часу. Колізії декількох PATCH запитів можуть бути більш небезпечними, ніж колізії PUT запитів, тому що деяким форматам змін необхідно виконуватися від відомої базової точки або ресурс буде пошкоджений. Клієнти, які використовують такий тип змін, повинні використовувати умовний запит на перевірку зміни ресурсу з моменту останнього доступу клієнта до нього. Наприклад клієнт може використовувати ETag у заголовку If-Match у самому PATCH запиті.

POST запит найчастіше використовується для створення нових ресурсів. Насправді він використовується до створення вкладених ресурсів. Іншими словами, при створенні нового ресурсу, POST запит відправляється до батьківського ресурсу і, таким чином, сервіс бере на себе відповідальність на

встановлення зв'язку ресурсу, що створюється з батьківським ресурсом, призначення новому ресурсу ID і т.п.

При успішному створенні ресурсу повертається HTTP код 201, а також у заголовку Location передається адреса створеного ресурсу.

POST не є безпечним чи ідемпотентним запитом. Тому рекомендується його використання для не ідемпотентних запитів. В результаті виконання ідентичних запитів POST надаються дуже схожі, але не ідентичні дані.

DELETE запит дуже простий для розуміння. Він використовується для видалення ресурсу, ідентифікованого конкретним URI (ID).

При успішному видаленні повертається 200 (OK) код HTTP, спільно з тілом відповіді, що містить дані віддаленого ресурсу (негативно позначається на економії трафіку) або загорнуті відповіді (Див. "Повертаються дані"). Також можливе використання коду HTTP 204 (NO CONTENT) без тіла відповіді.

Відповідно до специфікації HTTP, DELETE запит ідемпотентний. Якщо ви виконуєте DELETE запит на ресурс, він видаляється. Повторний DELETE запит до ресурсу закінчиться: ресурс видалено. Якщо DELETE запит використовується для декремента лічильника, DELETE запит не є демпотентним. Використовуйте POST для неідемпотентних операцій.

Тим не менш, існує застереження щодо ідемпотентності DELETE. Повторний DELETE запит до ресурсу часто супроводжується 404 (NOT FOUND) кодом HTTP через те, що ресурс вже видалений (наприклад з бази даних) і більше не доступний. Це робить DELETE операцію не ідемпотентною, але це загальноприйнятий компроміс на той випадок, якщо ресурс був вилучений з бази даних, а не помічений як віддалений.

Зверніть увагу на той факт, що специфікація HTTP не зобов'язує сервер розуміти всі методи (яких насправді набагато більше, ніж 4) обов'язковий тільки GET, а також не вказує серверу, що він повинен робити при отриманні запиту з тим чи іншим методом. Це означає, що сервер у відповідь на запит DELETE /index.php HTTP/1.1 не зобов'язаний видаляти сторінку index.php на

сервері, так само як на запит GET /index.php HTTP/1.1 не зобов'язаний повертати вам сторінку index.php, він може її видаляти.

Синхронізація: надішліть запит та зачекайте, поки сервер обробить дані та після повернить як завершиться обробка. Браузер клієнта нічого не може зробити протягом цього періоду.

Асинхронний: запит ініціюється обробкою подій – сервера (це все ще може зробити браузер) – обробка завершена

Найпоширенішим HTTP-запитом який і було використано при розробці є GET для перегляду веб-сторінок. Запит GET слід безпосередньо за URL-адресою і починається зі знака питання.

URL-адреса підтримує лише довжину близько 2 КБ, тобто кількість символів 2048. Коли запити AJAX виконуються за допомогою GET, сторінка, що відображається, буде неправильною. Загальний метод додає значення довільного параметра;

Значення у формі POST мають бути витягнуті та перетворені на рядки та пов'язані з амперсандами (аналогічно параметрам GET); обсяг даних, що відправляються становить 2 ГБ; ajax.setRequestHeader ('Content-Type', 'application/x-www-form-urlencoded'), який обробляє надіслані рядки; ajax.send (strings).

Синхронний та асинхронний запити можна розглянути за прикладом звичайного режиму B/S (синхронний), технологія AJAX (асинхронний) У методі ajax.open третій параметр встановлюється синхронний або асинхронний. Бібліотеки JS, такі як prototype, зазвичай за замовчуванням асинхронні, тобто мають значення true.

Режим синхронного виконання означає, що в режимі синхронного виконання оператор завжди контролюватиме потік програми до кінця програми. Наприклад, після операції запиту прикладна програма на клієнті чекатиме, поки сервер поверне результат запиту клієнту після видачі інструкції операції запиту на сервер, перш ніж перейти до наступної операції.

Як усі ми знаємо, програма потребує дуже багато часу для видалення всіх записів з великої таблиці. Якщо програма використовує однопоточний метод синхронного виконання, певна робота з видалення може затримати завершення іншої важливої роботи. Якщо програма очікує віддалену задачу, збій віддаленого сервера або мережевий збій або деякі непередбачувані умови можуть призвести до того, що програма чекатиме невизначено довго, що є найбільшим недоліком синхронного виконання.

Але режим синхронного виконання може спростити складність програмування. Програмісти можуть використовувати режим синхронного виконання ODBC або використовувати елементи керування даними та змінні об'єкти бази даних для написання програм без особливого розуміння використання більш складних API-інтерфейсів ODBC 2.0, які можуть підвищити ефективність розробки, але програма працює швидше, ніж не так швидко, як у режимі асинхронного виконання

Режим асинхронного виконання означає, що режим асинхронного виконання порядок виконання кожного оператора не обов'язково збігається з порядком виконання операторів. Наприклад, для операції запиту програма клієнта надсилає інструкцію операції запиту на сервер і негайно виконує наступну інструкцію інструкції оператора запиту, не чекаючи, поки сервер поверне результат запиту клієнту. Асинхронне виконання дозволяє звільнити програми від обмежень одного завдання, що підвищує гнучкість та ефективність виконання програм. Однак є деякі проблеми з асинхронним режимом виконання, наприклад він збільшує складність програмування, особливо при написанні програм, які вимагають високої функціональної сумісності.

У завантаженій клієнт-серверній системі підходить режим асинхронного виконання. У цьому середовищі тимчасові затримки є частими і тривалими, і витрати асинхронного виконання проти незначні. Однак, якщо середовище, в якому виконується програма, є більш складним, необхідно створити повний механізм для періодичної перевірки стану виконання функції

визначення наступного плану виконання. Є багато способів перевірити період, наприклад, встановити таймер у додатку та обробити інформацію WM_TIMER.

Хоча асинхронний режим виконання дуже складний у програмуванні, може забезпечити паралельне виконання кількох завдань, що значно підвищує ефективність виконання.

Вибір та налаштування режиму виконання Вибір синхронного або асинхронного режиму у розробці програм є більш складним рівнем. Коли запит або модифікація до бази даних є відносно простою, синхронний режим виконання є гарним вибором: він може повернути дані результату за кілька секунд або менше. Крім того, коли програма не може продовжити виконання, доки не буде отримано набір результатів, зовсім не обов'язково використовувати режим асинхронного виконання. У разі складних запитів, особливо операцій UPDATE або DELETE складних багаторядкових баз даних, виконання може тривати багато часу, і потрібно асинхронний режим виконання, щоб дозволити користувачам одночасно керувати іншими частинами програми.

Контролера, який відповідає за авторизування та реєстрацію нового користувача, можна побачити у коді UserController.

Розглянемо код UserController на рисунку 3.4

```
@RestController
@RequestMapping("/user")
@AllArgsConstructor
public class UserController {

    private UserService userService;
    private ModelMapper modelMapper;

    @PostMapping
    public ResponseEntity registerUser(@RequestBody User json) {
        int statusCode = 400;
        if (userService.save(json))
            statusCode = 200;
        return ResponseEntity
                .status(statusCode)
                .body(json);
    }

    @PostMapping("login")
    public ResponseEntity login(@RequestBody User user) {
        User ownUser = userService.getByUsername(user.getUsername());
        int statusCode = 400;
        if (user.getPassword().equals(ownUser.getPassword())) {
            statusCode = 200;
        }
        return ResponseEntity
                .status(statusCode)
                .body(user);
    }
```

Рисунок 3.4 – Лістинг коду UserController

Для роботи з базою даних необхідно було створити також сервіс, який буде відповідати за збереження та редагування даних, що можна побачити на у коді UserServiceImpl. Розглянемо Код сервісу UserServiceImpl на рисунку 3.5.

```
@AllArgsConstructor
public class UserServiceImpl implements UserService {

    private UserRepository userRepository;

    @Override
    public boolean save(User user) {
        if (userRepository.findAllByEmailOrUsername(user.getEmail(), user.getUsername()).isEmpty()) {
            userRepository.save(user);
            return true;
        } else {
            return false;
        }
    }

    @Override
    public User getById(Long id) { return userRepository.getOne(id); }

    @Override
    public Collection<User> getAll() { return userRepository.findAll(); }

    @Override
    public User edit(User user) { return userRepository.saveAndFlush(user); }

    @Override
    public void delete(User user) { userRepository.delete(user); }

    @Override
    public User getByUsername(String username) {
        return userRepository.getByUsername(username);
    }
}
```

Рисунок 3.5 – Лістинг коду UserService

Для підключення до бази даних необхідно також змінити необхідні параметри, підключити драйвер та вказати адресу url. . Розглянемо код application.properties на рисунку 3.6

```
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/quizel
spring.datasource.username=postgres
spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update
spring.datasource.type=com.zaxxer.hikari.HikariDataSource

# JPA Configuration
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.generate-ddl=true
```

Рисунок 3.6 – Лістинг коду application.properties

Для того щоб при надсиланні результатів запиту в форматі JSON всі дані надсилались вірно та була виключена рекурсія потрібно створити адаптер. Розглянемо код адаптера на рисунку 3.7

```
@Service
public class AnswerAdapter implements JsonSerializer<Answer> {
    @Override
    public JsonElement serialize(Answer answer, Type type, JsonSerializationContext jsonSerializationContext
        JsonObject jsonObject = new JsonObject();
        jsonObject.addProperty( property: "id", answer.getId());
        jsonObject.addProperty( property: "answer", answer.getAnswer());
        jsonObject.addProperty( property: "right_answer", answer.getRight_answer());
        jsonObject.addProperty( property: "quiz_id", answer.getQuize().getId());
        return jsonObject;
    }
}
```

Рисунок 3.7 – Лістинг коду адаптера

Щоб нормалізувати таблицю зберігаючи відповіді і знизити надлишковість інформації була створена таблиця Quize для збереження запитань маючи зв’язок “Багато до одного”. Розглянемо сутність Quize на рисунку 3.8

```java
@Entity
@Setter
@Getter
@Table(name = "quize")
public class Quize {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String question;

    @OneToMany(fetch = FetchType.EAGER, mappedBy = "quize")
    private Set<Answer> answers;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "users")
    private User users;

    @Override
    public String toString() {
        return "Quize{" +
                "id=" + id +
                ", question='" + question + '\'' +
                ", users=" + users +
                '}';
    }
}
```

Рисунок 3.8 – Лістинг коду Quize

Для зберігання відповідей була створена окрема табличка та зв’язана із таблицею Quize, рисунок 3.9

```java
@Table(name = "answers")
@Entity
@Getter
@Setter
public class Answer {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String answer;
    private Integer right_answer;
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "quize")
    private Quize quize;

    public Answer() {
    }

    public Answer(String answer, Quize quize, Integer right_answer) {


        this.answer = answer;

        this.quize = quize;

        this.right_answer = right_answer;

@Override
public String toString() {
    return "Answer{" +
            "id=" + id +
            ", answer='" + answer + '\'' +
            '}';
}

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

    Answer answer1 = (Answer) o;

    if (id != null ? !id.equals(answer1.id) : answer1.id != null) return false;
    return answer != null ? answer.equals(answer1.answer) : answer1.answer == null;
}
```

Рисунок 3.9 – Лістинг коду Answer

Найчастіше, в клієнт-серверних додатках дані на клієнті (шар представлення) і на сервері (шар предметної області) структуруються по-різному. На стороні сервера це дає нам можливість комфортно зберігати дані в базі даних або оптимізувати використання даних для продуктивності, в той же час займатися “user-friendly” відображенням даних на клієнті, і, для серверної частини, потрібно знайти спосіб як перекладати дані з одного формату в інший. Звичайно, існують і інші архітектури додатків, але ми зупинимося на поточному як спрощення. DTO-подібні об'єкти можуть

використовуватись між будь-якими двома шарами представлення даних, рисунок 3.10

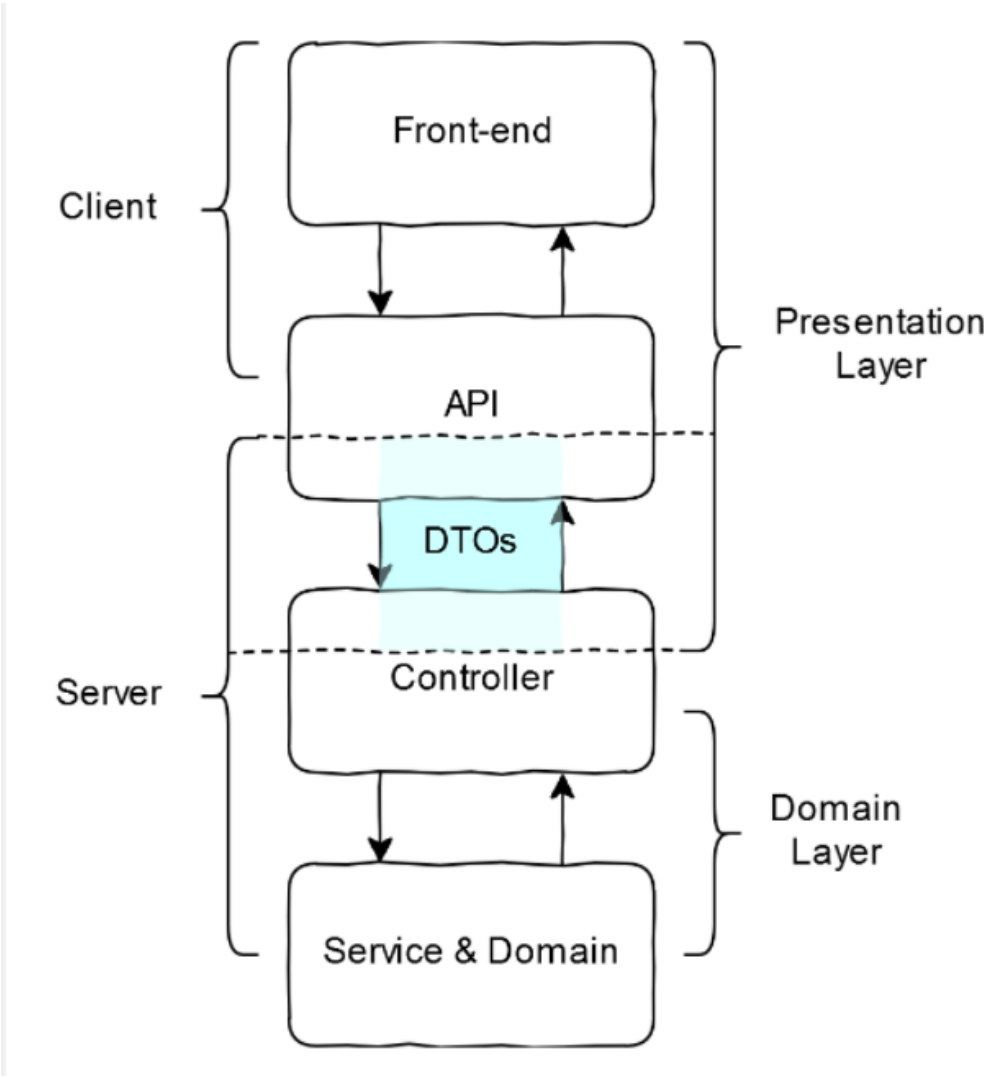



Рисунок 3.10 – DTO схема

По-перше, дуже важливо розуміти, що ви не повинні використовувати DTO. Це насамперед патерн і ваш код може працювати чудово і без нього.

Якщо ви використовуєте одне представлення даних на обидва шари, ви можете використовувати ваші сутності як DTO.

Якщо ви хочете займатися серіалізацією ваших сутностей в JSON, то я не можу вас зупинити!

Вони також допомагають документувати шар уявлення в людині, що читається. Мені подобається використовувати DTO і, я думаю, ви теж могли б їх використовувати, адже це також сприяє зменшенню зачеплення (decoupling)

між шаром уявлення і предметним шаром, дозволяючи застосуванню бути більш гнучким і зменшуючи складність його подальшої розробки.

Тим не менш, не всі DTO є добрими. Хороші DTO допомагають створювати API згідно з найкращими практиками та відповідно до принципів чистого коду.

Вони повинні дозволяти розробникам писати API, яке внутрішньо узгоджено. Опис параметра на одній із кінцевих точок (endpoint) має застосовуватися і до параметрів з тим самим ім'ям на всіх пов'язаних точках. Як приклад, візьмемо вищенаведений фрагмент коду. Якщо поле price при запиті визначено як “ціна з ПДВ”, то й відповідь визначення поля price не має змінитись. Узгоджене API запобігає помилкам, які могли виникнути через відмінності між кінцевими точками, і водночас полегшує введення нових розробників у проект.

DTO повинні бути надійними і мінімізувати необхідність написання шаблонного коду.

В своєму додатку я також застосував DTO наприклад для класу Answer на рисунку 3.11


Рисунок 3.11 – Answer DTO

На рисунку 3.12 можна побачити QuizeDTO

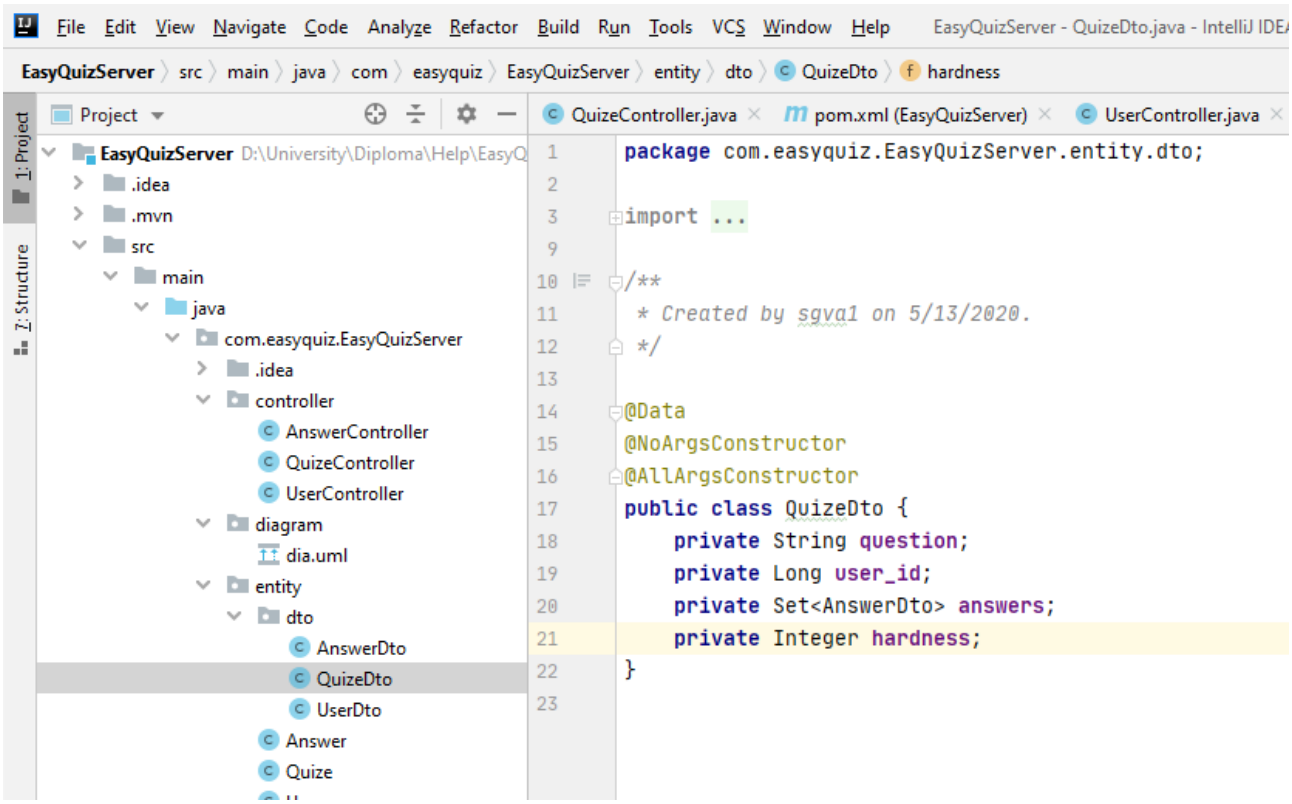



Рисунок 3.12 – Quize DTO

На рисунку 3.13 можна побачити UserDTO

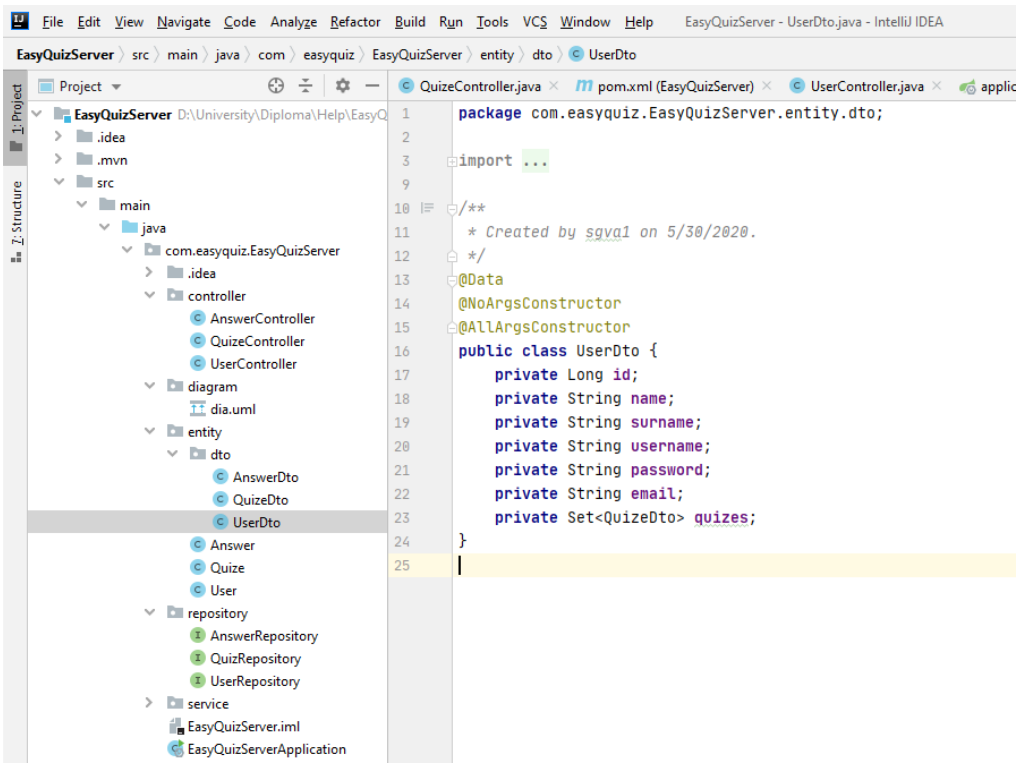



Рисунок 3.13 – User DTO

На рисунку 3.14 можна побачити вікно створення тестів та поле чекбоксу для автоматичної системи коригування рівня складності запитань

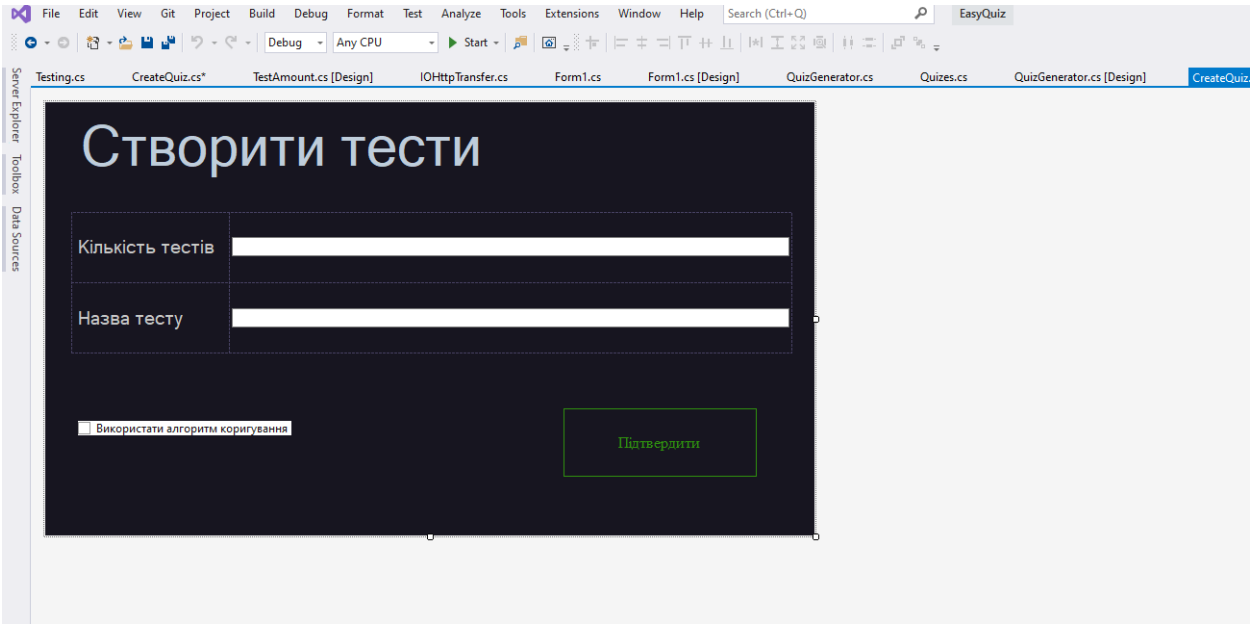



Рисунок 3.14 – Вікно створення тестів

На рисунку 3.15 можна побачити алгоритм що автоматично коригує рівень складності запитань

```
                // Console.Write(" {0}", i);
            }
        }

        4 references
        void herewego(List<Quizes> listfirst, int number) {


            if (easyQuest == true)

            {
                for (int i = number; i < listfirst.Capacity; i++) {
                    if (listfirst[i].Hardness < 3)
                    {

                        label5.Text = listfirst[i].Question;
                        List<Answer> listanswerfirst = new List<Answer>(listfirst[i].Answers);
                        right_answer = listanswerfirst[0].Right_answer;
                        char[] charsToTrim = { '*', ' ', '\'' };

                        label6.Text = Convert.ToString(listanswerfirst[0].AnswerSet);
                        label9.Text = Convert.ToString(listanswerfirst[1].AnswerSet);
                        label10.Text = Convert.ToString(listanswerfirst[2].AnswerSet);
                        label11.Text = Convert.ToString(listanswerfirst[3].AnswerSet);
                        easyQuest = false;
                        break;
                    }


                }


            }
            else {
```

Рисунок 3.15 – Алгоритм автоматичного коригування складністю

Щоб мати змогу працювати з даними та об’єктами переносячи їх в базу даних і назад використовувались Spring Data

Spring Data – додатковий зручний механізм для взаємодії з сутностями бази даних, організації їх у репозиторії, вилучення даних, зміна, у яких випадках для цього достатньо оголосити інтерфейс і метод в ньому, без імплементації. Основне поняття у Spring Data – це репозиторій. Це кілька інтерфейсів, які використовують JPA Entity для взаємодії з нею.

Запити до сутності можна будувати з імені методу. Для цього використовується механізм префіксів find…By, read…By, query…By, count…By, та get…By, далі від префіксу методу починає розбір решти. Вступна пропозиція може містити додаткові вирази, наприклад, Distinct. Далі перший By діє як роздільник, щоб вказати на початок фактичних критеріїв. Можна визначити умови для властивостей сутностей та об'єднати їх за допомогою And та Or. Далі поговоримо звідки з’явилась Spring Data.

JPA – це абревіатура Java Persistence API. Його суть – специфікація ORM (Object Relational Mapping, що описує відносини між об'єктами та таблицями). Хоча не реалізує конкретну структуру ORM, він визначає стандарт всім структур ORM. API надає низку інтерфейсів. Очевидно, що інтерфейси не можна використовувати безпосередньо. Хоча можна сказати, що програмування, орієнтоване на інтерфейси, не сприяє трансплантації та розширенню проекту, якщо кожен розробник розмовляє сам із собою, щоб створити набір реалізацій. Отже, найбільший внесок JPA - це реалізація ORM. Надає єдину специфікацію (рисунок 3.10), що включає Hibernate, TopLink і т.д.

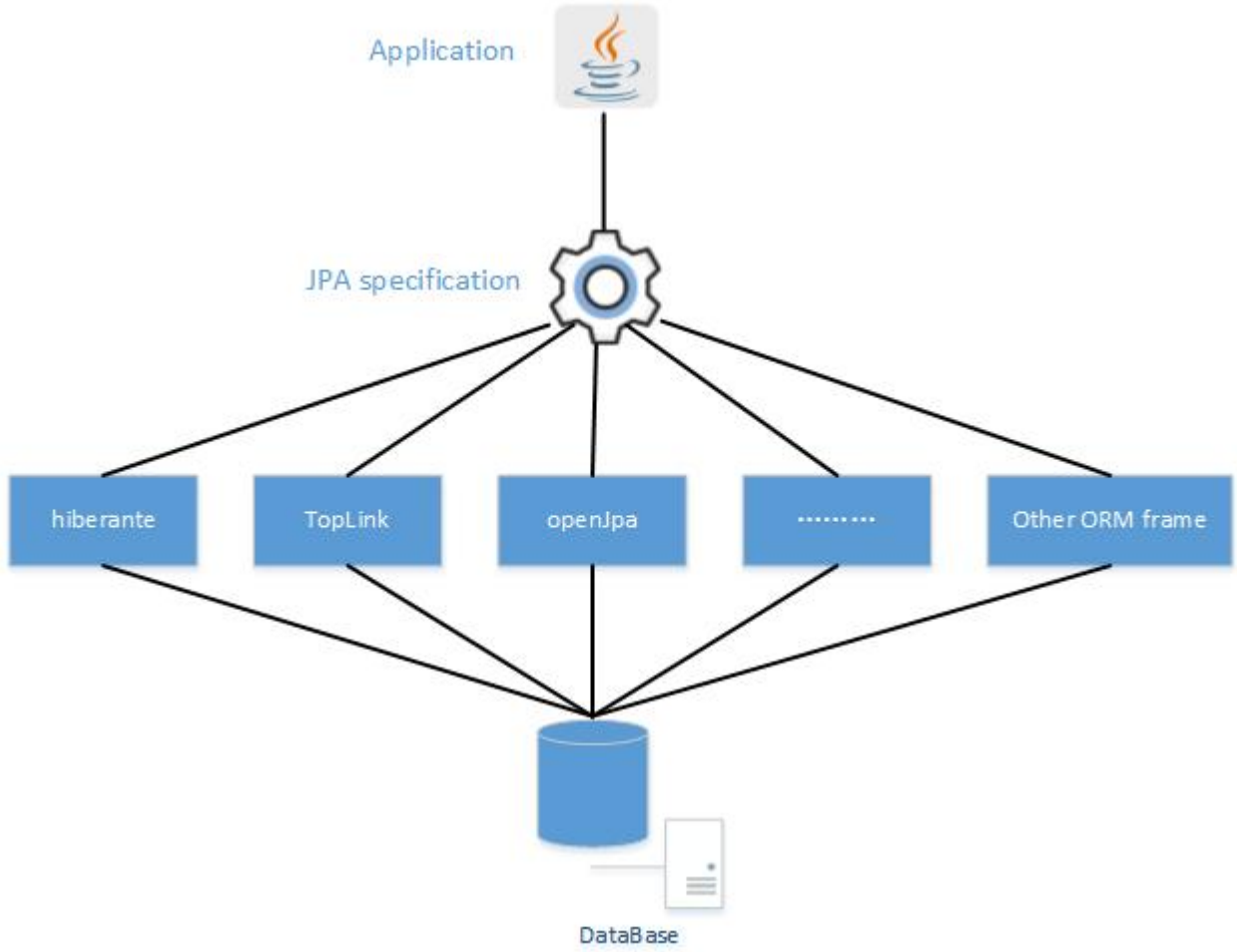



Рисунок 3.10 – Jpa специфікація

Раніше Spring не сильно допомагав у бізнес-логіці рівня персистентності, але після появи Spring Data JPA бізнес-розробка персистентності різних класів сутностей також стала простішою. Spring Data JPA Забезпечує реалізацію рівня репозиторію згідно зі специфікацією JPA., Але Spring не вказує, яку структуру ORM використовувати, тому розробникам все одно потрібно вирішувати самим.

Найбільша зручність це приносить: хоча ORM реалізував специфікацію JPA, різниця між різними платформами, як і раніше, робить переключення дорогим (переписування постійного коду бізнес-логіки), Spring Data JPA дозволяє нам уникнути цієї частини витрат. Єдине, що потрібно зробити, це оголосити інтерфейс рівня збереження.

## 3.4 Висновки

У третьому розділі було обґрунтовано вибір мов, технологій програмування та IDE, які були використані при створенні даного програмного додатку EasyQuiz. Було розроблено програми взаємодії програмних модулів, а також блок авторизації.

# 4 ТЕСТУВАННЯ ПРОГРАМИ

## 4.1 Тестування програмного забезпечення

Тестування програмного забезпечення (англ. Software Testing) – це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки:

- Тестування дозволяє перевірити, чи правильно реалізовано усі вимоги до ПЗ, що розроблялось.
- Тестування допомагає у виявленні дефектів / помилок та забезпечує їх розпізнавання / вирішення до етапу розгортання програмного забезпечення.
- Тестування пом’якшує наслідки та ризики втрат якщо програмний продукт все ж випустили по неправильних вимогах. Вимоги в такому випадку намагаються частково виправити, переоцінити. Програму покращити.
- Тестування також демонструє, що створене ПЗ працює відповідає також вимогам до продуктивності.
- Тестування допомагає перевірити належну інтеграцію та взаємодію програми з навколишнім середовищем[28].

На сьогодні тестування програмного забезпечення – один з найбільш дорогих етапів життєвого циклу програмного забезпечення [29], на нього відводиться від 50% до 65% загальних витрат.

Один із загальних законів практичного програмування полягає в тому, що жодна програма не дає бажаних результатів при першій спробі трансляції та виконання. Певне уявлення про справжні причини появи помилок у роботі програми дає таке процентне співвідношення джерел збоїв:

- Вхідні дані – 1%.

- Помилки користувача – 5%.
- Апаратура – 1%.
- Системне програмне забезпечення – 3%.
- Розробка системи – 15%.
- Програмування – 75%.

Програміст повинен не тільки писати ефективні програми, але і знаходити в них усілякі помилки. Сучасна практика навчання програмуванню орієнтована, в основному, тільки на виконання програмістом першої половини своєї роботи. Це все одно, як навчати льотчика тільки зльоту, припускаючи, що з посадкою машини він якось розбереться сам, якщо буде виконувати всі операції зльоту в зворотному порядку.

Існують два типи програмних помилок:

- Синтаксичні помилки виникають через порушення правил мови програмування. Такі помилки зазвичай виявляються під час компіляції. Можуть бути виключені порівняно легко. Навіть якщо не переглядати текст програми можна бути впевненим, що компілятор на стадії трансляції знайде помилки і видасть відповідні попередження. Фактично пошук помилок здійснює компілятор, а їхнє виправлення - програміст;
- Семантичні (логічні) помилки ті, що призводять до некоректних обчислень або помилок під час виконання (run-time error). Семантичні помилки усувають зазвичай за допомогою виконання програми з ретельно підібраними перевірочними даними, для яких відома правильна відповідь[30].

Протестуємо роботу програмного додатку та спробуємо зареєструватись. Для початку потрібно запустити сервер PostgreSQL (рисунок 4.1). Далі запускаємо головну форму та спробуємо зареєструвати нового користувача.

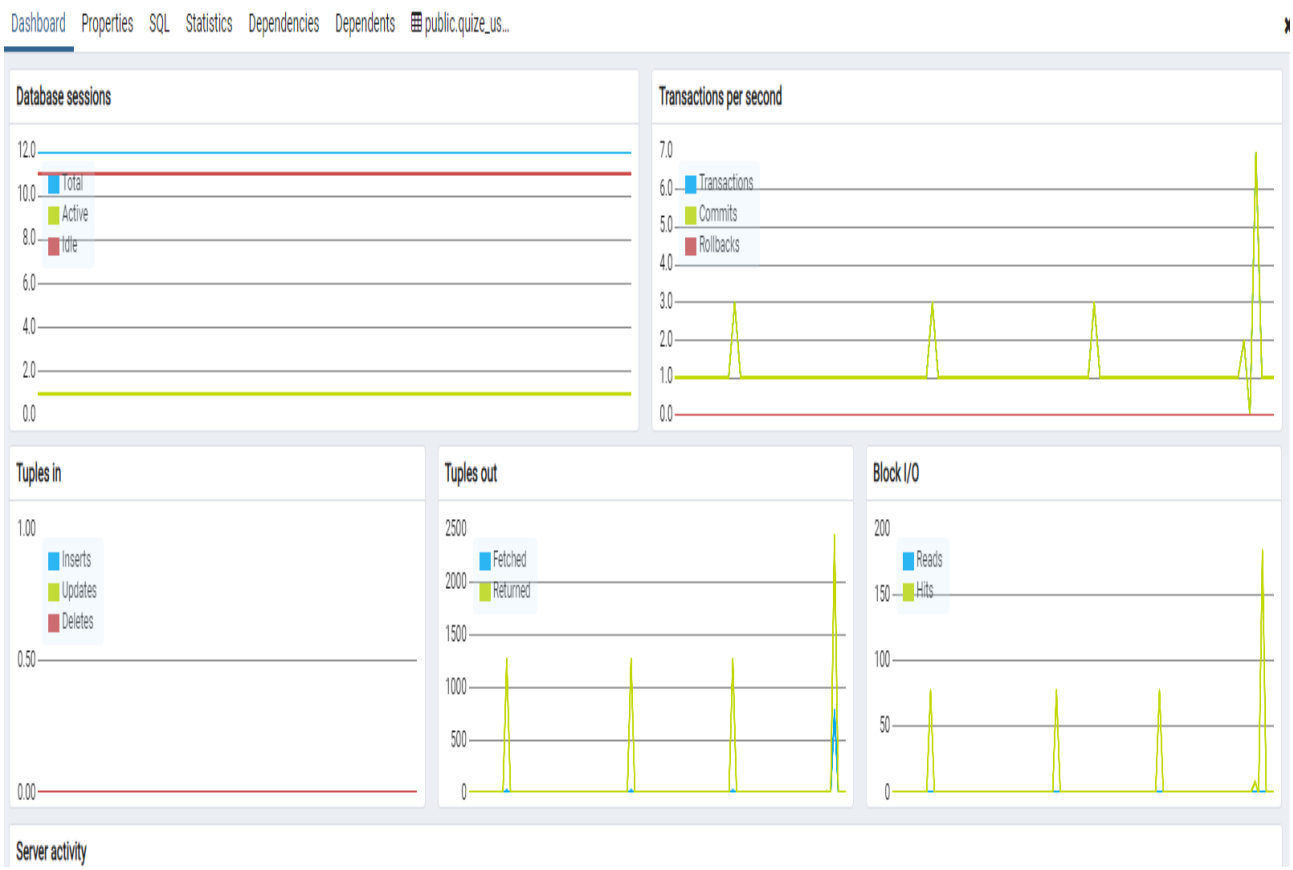



Рисунок 4.1 – Статус активності бази даних PostgreSQL

Заповнюємо поля для реєстрації відповідно до формату які вони вимагають. Всі введені символи порівнюються з ASCII таблицею і заповнюються лише ті які підходять під визначено маску, якщо ж символи їй не відповідають з'являється відповідне попередження (знак червоного кольору), що сигналізує про неправильний введений формат даних. Так наприклад для полів «Ім’я», «Прізвище» дозволяється введення українською, російською та англійською розкладками але забороняється числовий формат даних (рис. 4.2). Поля, які відповідають за «Е-мейл» та «Логін» користувача, приймають лише цифрову і англійську розкладку клавіатури забороняючи російську і українську відповідно. Поле, що призначене для введення паролю може приймати лише цифрові значення унеможливлюючи введення символів та літер (рисунок 4.3).

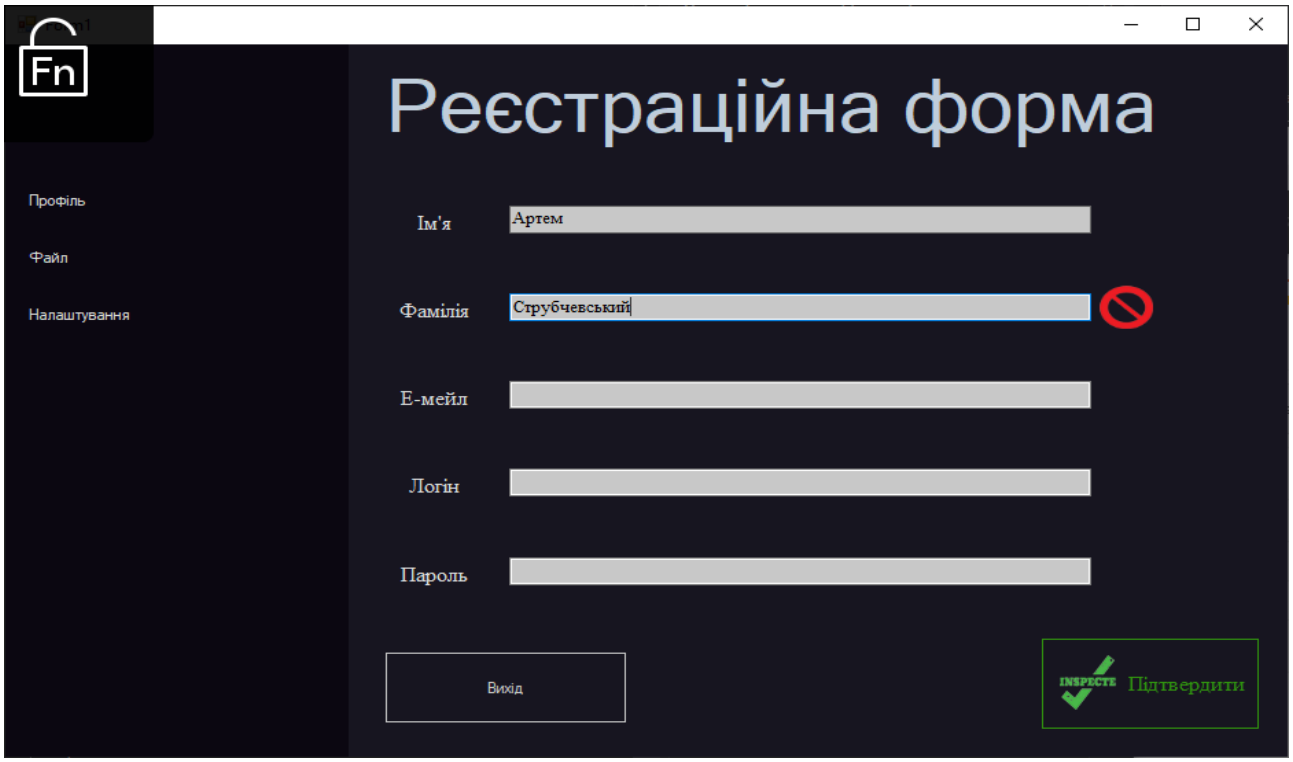



Рисунок 4.2 – Перевірка відповідності введених даних в поле «Прізвище»

Перевірка проходить в реальному часі і при кожному введені символу викликає метод перевірки на правильність формату введення.

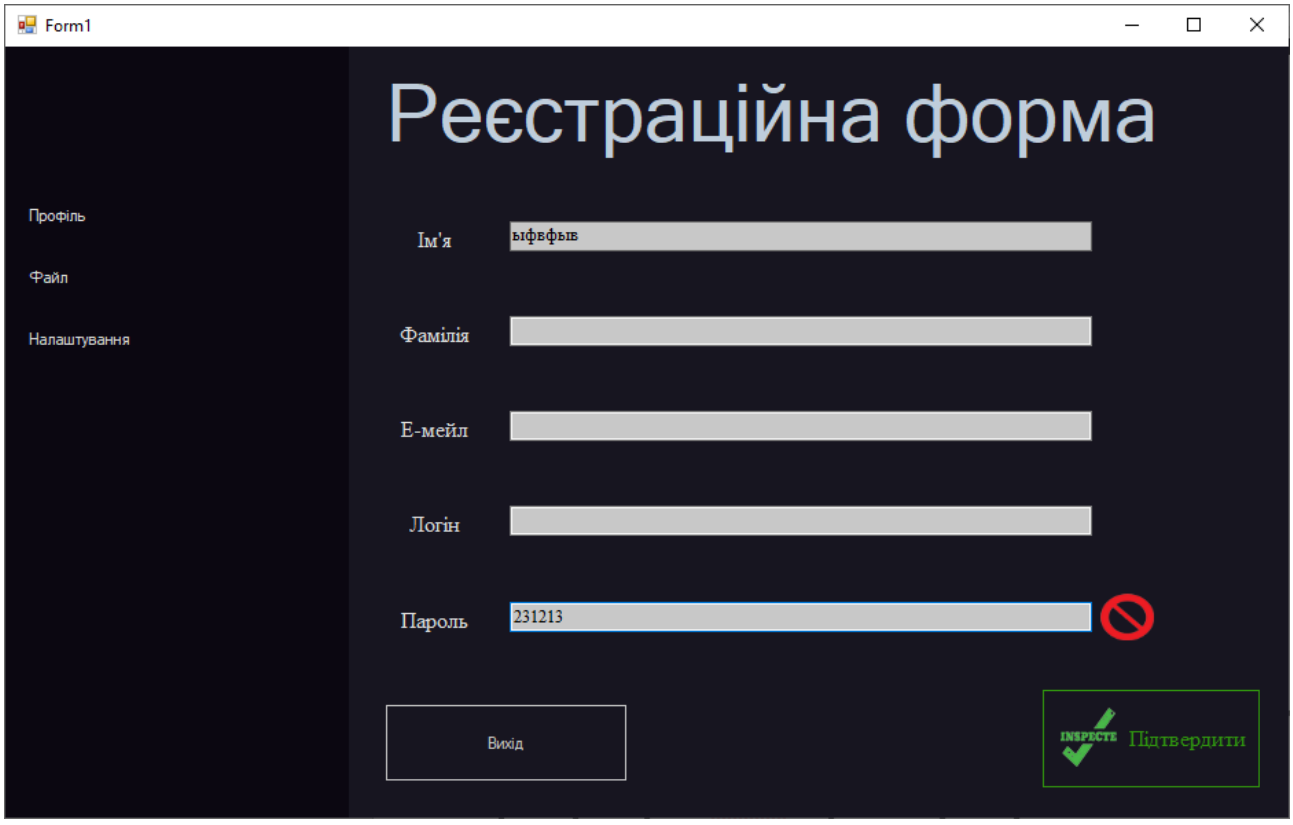



Рисунок 4.3 – Перевірка відповідності введених даних в поле «Пароль»

Для надсиалання запиту потрібно заповнити всі поля відповідно до формату якщо цього не зробити запити не буде надіслано і буде виведено повідомлення що вимагатиме дозаповнити всі не заповнені дані (рисунок 4.4).

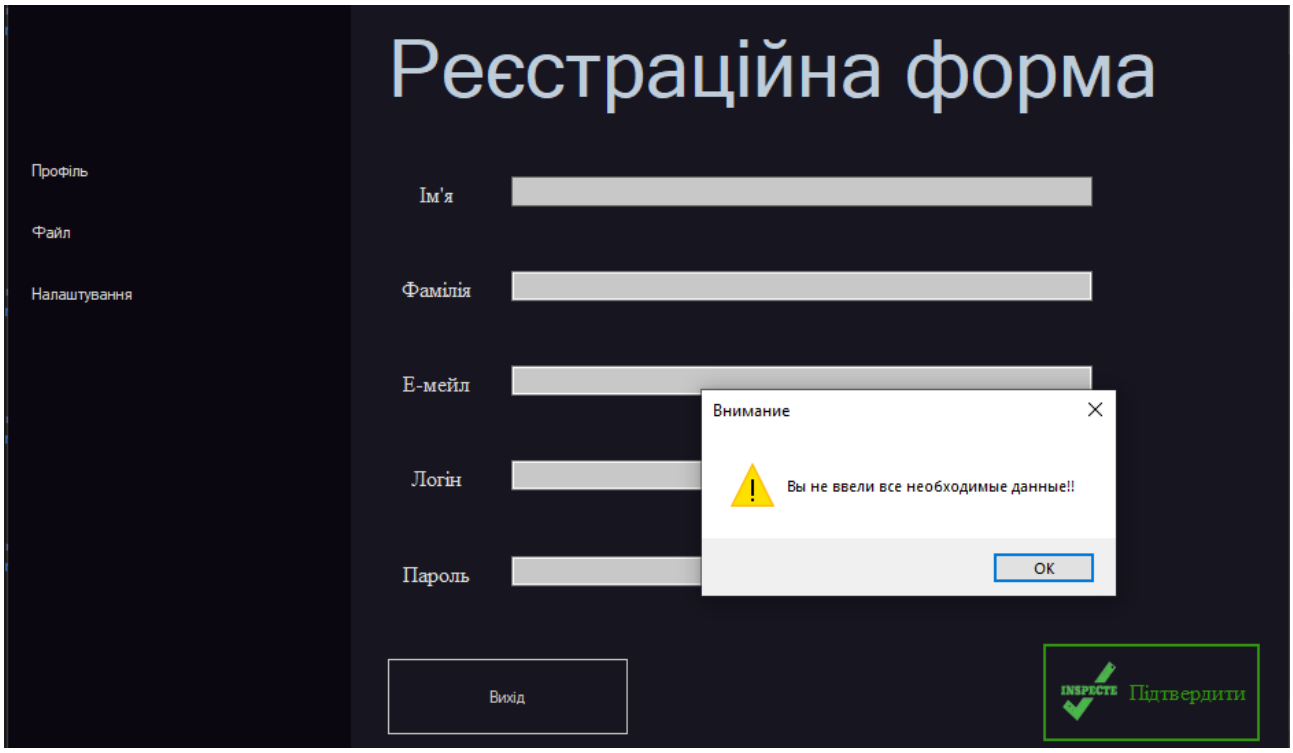



Рисунок 4.4 – Перевірка на заповнення всіх полів

Після проведення всіх необхідних перевірок запит надсилається до нашого контролеру в серверній частині проекту та всі дані записуються в базу, реакцію на запит можемо бачити на рисунку 4.5.

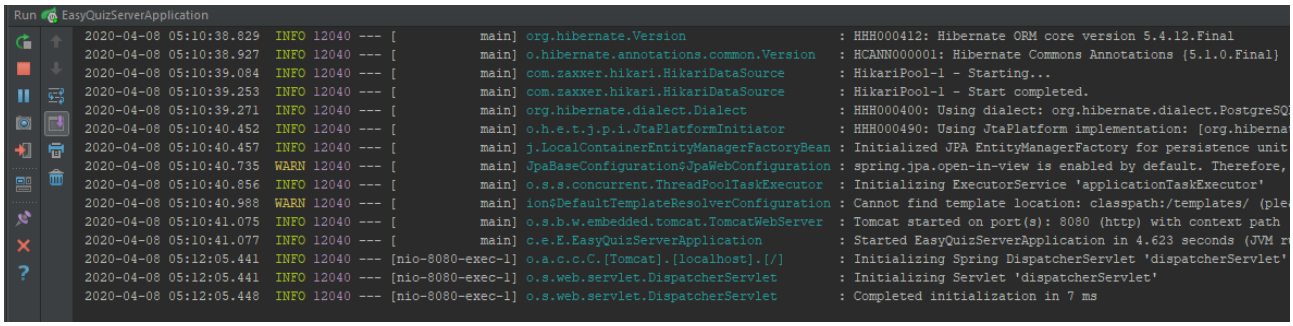



Рисунок 4.5 – Стрічка стану всіх запитів до серверу в IDE Intellij Idea

Більш детально перевірити та протестувати Spring частину проекту можна за допомогою такої програми як Postman (рисунок 4.6). Він дає змогу

надсилати http запити, інформацію в форматі json допомагаючи прослідкувати реакцію на введені дані, а також спрощує знаходження помилок в додатках [21].

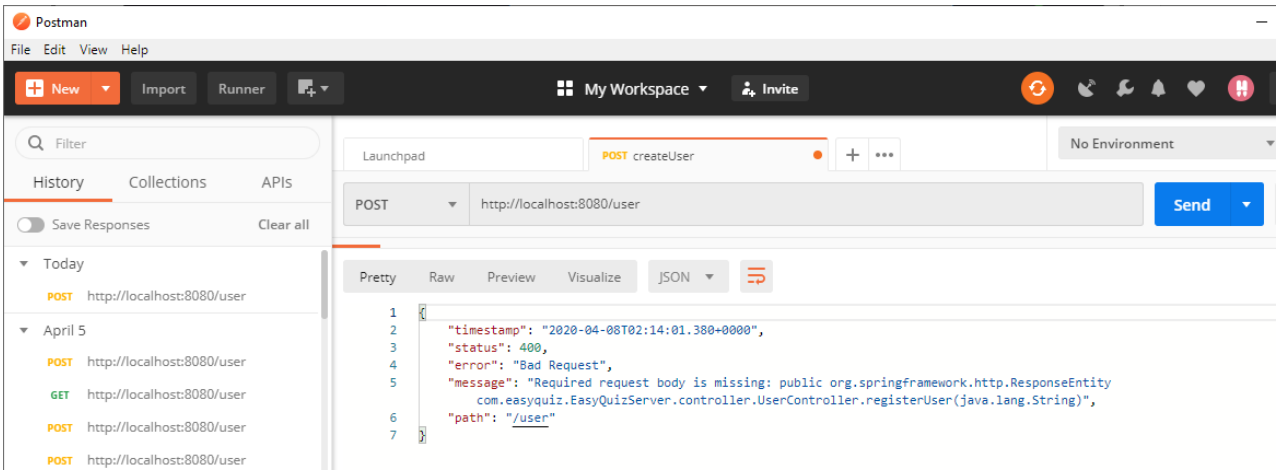



Рисунок 4.6 – Запит Postman

Після виконання усіх перевірок та запису на сервер інформації додаток отримавши позитивну відповідь від серверу відобразить позначки зеленого кольору, що означатиме успішну реєстрацію нового користувача (рисунок 4.7).

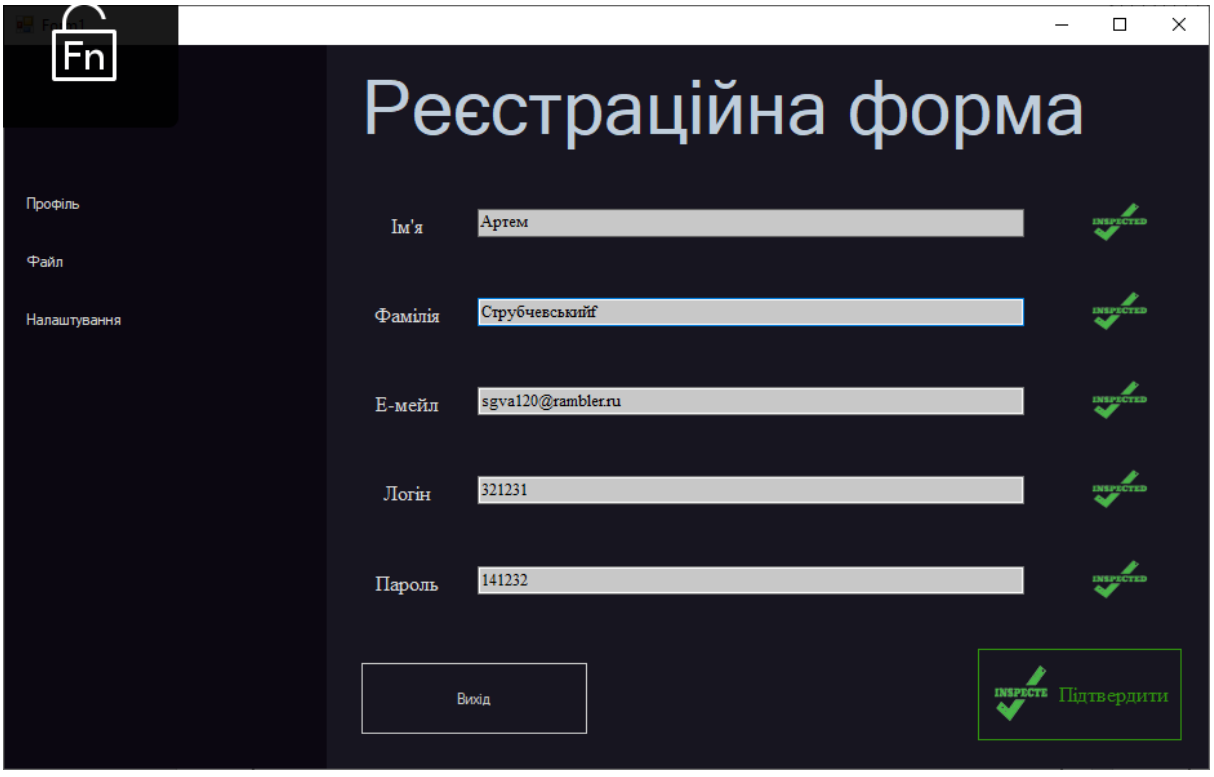



Рисунок 4.7 – Успішна реєстрація нового користувача

Також на рисунку 4.8 можемо бачити завантажене вікно тестування та отриманні із серверу питання та відповіді.

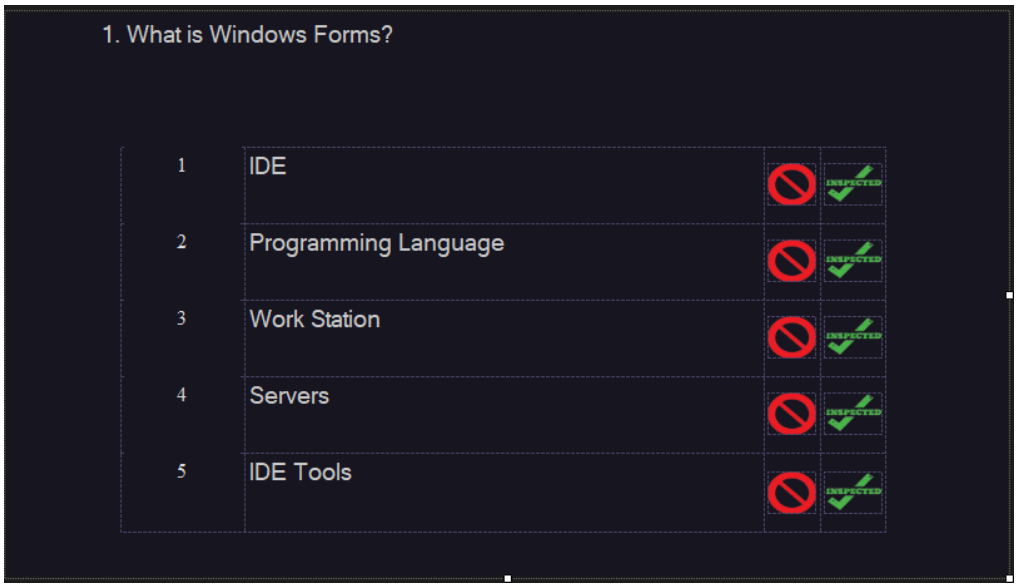



Рисунок 4.8 – Вікно тестування

За допомогою алгоритму можемо перевірити введену відповідь на вірність (рисунок 4.9).

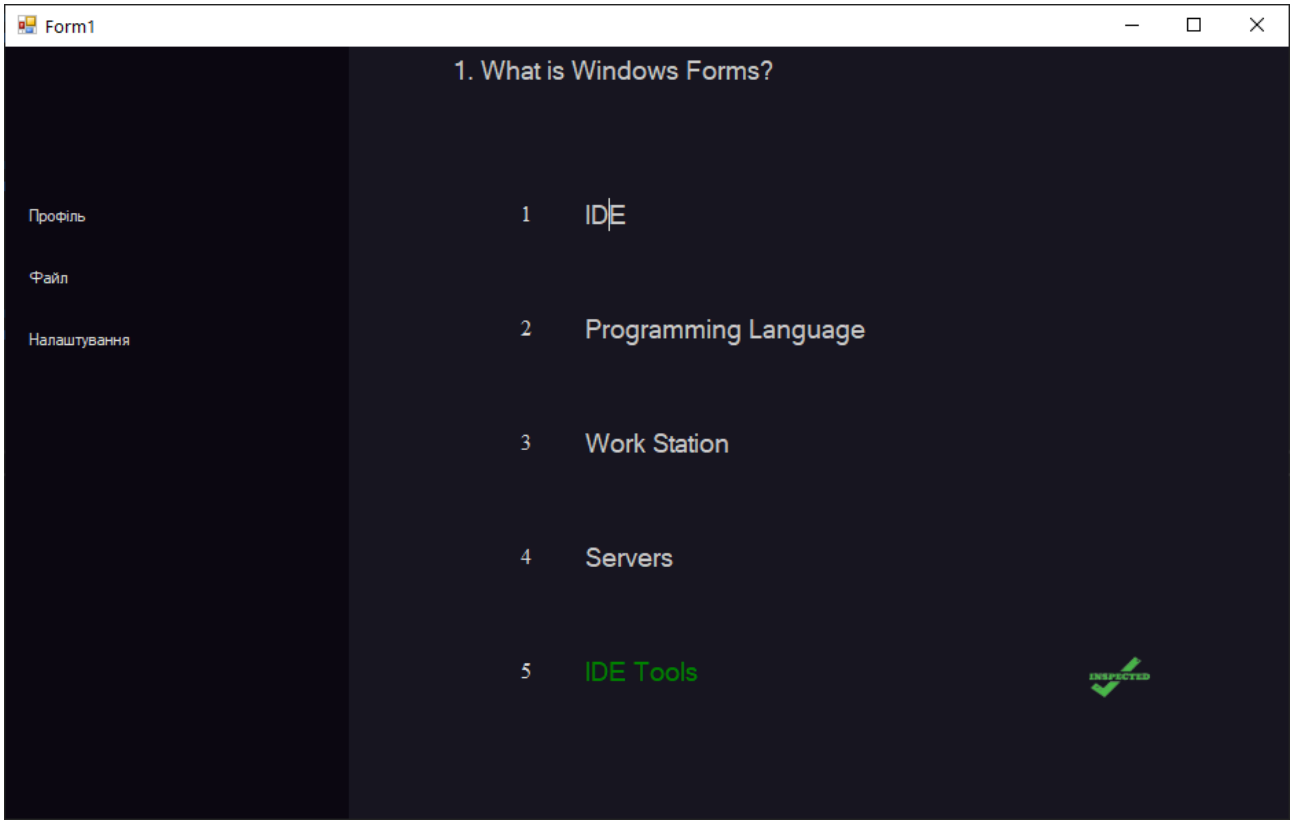



Рисунок 4.9 – Перевірка на правильність введення відповіді

Отже, розроблений програмний додаток для індивідуальної системи тестування працює вірно та проходить всі необхідні перевірки.

## 4.2 Розробка інструкції користувача

Вимоги до складу і параметрів технічних засобів. Система повинна працювати на IBM-сумісних персональних комп'ютерах. У таблиці 4.1 та 4.2 наведено мінімальну конфігурацію персонального комп'ютера для запуску програми.

Таблиця 4.1 – Рекомендована конфігурація:

| | |
|---|---|
| Тип процесора | Intel i3 з частотою 1,7 ГГц і вище; |
| Об'єм оперативної пам'яті | 2 ГБ |
| Розмір жорсткого диску | 20 ГБ |
| Графічний пристрій | графічний пристрій DirectX9 |

Розмір дискового простору, що займає програма: 2,6 Мбайт.

Розмір оперативної пам'яті, що займає програма: 1,5 МБайт.

Програма працює під управлінням операційної системи Windows (XP/Vista/7/8/8.1,10).

Для роботи програми необхідно мати доступ до інтернету та налаштований сервер до якого будуть надходити необхідні запити оброблятись та зберігати в базі даних.

## 4.3 Висновки

Було протестовано працездатність програмного додатку EasuQuiz, масок для введення, надсилання запитів на сервер Spring Java, зв'язок з базою даних PostgreSql, що продемонструвало відповідність додатку до усіх поставлених завдань.

# 5 ЕКОНОМІЧНА ЧАСТИНА

## 5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу методу та програмного засобу діагностики знань з автоматичним коригуванням рівня складності запитань.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів: Гулько Сергій Віталійович (компанія Айбекс айті на посаді Team Lead DA підрозділу), Оверчук Анна Вікторівна (компанія Pillar на посаді QA ), Куценко Дмитро Геннадійович (компанія Pillar на посаді ОВТ). Для проведення технологічного аудиту було використано таблицю 5.1 [1] в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

| Критерії оцінювання та бали (за 5-ти бальною шкалою) | | | | | |
|---|---|---|---|---|---|
| Кри-терій | 0 | 1 | 2 | 3 | 4 |
| Технічна здійсненність концепції: | | | | | |
| 1 | Достовірність концепції не підтверджена | Концепція підтверджена експертними висновками | Концепція підтверджена розрахунками | Концепція перевірена на практиці | Перевірено роботоздатність продукту в реальних умовах |
| Ринкові переваги (недоліки): | | | | | |
| 2 | Багато аналогів на малому ринку | Мало аналогів на малому ринку | Кілька аналогів на великому ринку | Один аналог на великому ринку | Продукт не має аналогів на великому ринку |
| 3 | Ціна продукту значно вища за ціни аналогів | Ціна продукту дещо вища за ціни аналогів | Ціна продукту приблизно дорівнює цінам аналогів | Ціна продукту дещо нижче за ціни аналогів | Ціна продукту значно нижче за ціни аналогів |
| 4 | Технічні та споживчі властивості продукту значно гірші, ніж в аналогів | Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів | Технічні та споживчі властивості продукту на рівні аналогів | Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів | Технічні та споживчі властивості продукту значно кращі, ніж в аналогів |

Продовження табл. 5.1

| | | | | | |
|---|---|---|---|---|---|
| 5 | Експлуатаційні витрати значно вищі, ніж в аналогів | Експлуатаційні витрати дещо вищі, ніж в аналогів | Експлуатаційні витрати на рівні експлуатаційних витрат аналогів | Експлуатаційні витрати трохи нижчі, ніж в аналогів | Експлуатаційні витрати значно нижчі, ніж в аналогів |
| Ринкові перспективи | | | | | |
| 6 | Ринок малий і не має позитивної динаміки | Ринок малий, але має позитивну динаміку | Середній ринок з позитивною динамікою | Великий стабільний ринок | Великий ринок з позитивною динамікою |
| 7 | Активна конкуренція великих компаній на ринку | Активна конкуренція | Помірна конкуренція | Незначна конкуренція | Конкурентів немає |
| Практична здійсненність | | | | | |
| 8 | Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї | Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців | Необхідне незначне навчання фахівців та збільшення їх штату | Необхідне незначне навчання фахівців | Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї |
| 9 | Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні | Потрібні незначні фінансові ресурси. Джерела фінансування відсутні | Потрібні значні фінансові ресурси. Джерела фінансування є | Потрібні незначні фінансові ресурси. Джерела фінансування є | Не потребує додаткового фінансування |
| 10 | Необхідна розробка нових матеріалів | Потрібні матеріали, що використовуються у військово-промисловому комплексі | Потрібні дорогі матеріали | Потрібні досяжні та дешеві матеріали | Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві |
| 11 | Термін реалізації ідеї більший за 10 років | Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років | Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років | Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років |
| 12 | Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на | Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу | Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу | Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту | Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту |

| | | | | | |
|---|---|---|---|---|---|
| | виробництво та реалізацію продукту | | | | |

Таблиця 5.2 – Рівні комерційного потенціалу розробки

| Середньоарифметична сума балів СБ, розрахована на основі висновків експертів | Рівень комерційного потенціалу розробки |
|---|---|
| 0-10 | Низький |
| 11-20 | Нижче середнього |
| 21-30 | Середній |
| 31-40 | Вище середнього |
| 41-48 | Високий |

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

| Критерії | Прізвище, ініціали, посада експерта | | |
|---|---|---|---|
| | Гулько Сергій Віталійович | Оверчук Анна Вікторівна | Куценко Дмитро Геннадійович |
| | Бали, виставлені експертами: | | |
| 1 | 3 | 4 | 4 |
| 2 | 4 | 4 | 4 |
| 3 | 1 | 2 | 1 |
| 4 | 2 | 3 | 3 |
| 5 | 3 | 3 | 3 |
| 6 | 1 | 1 | 1 |
| 7 | 4 | 5 | 5 |
| 8 | 3 | 3 | 3 |
| 9 | 2 | 1 | 3 |
| 10 | 4 | 4 | 5 |
| 11 | 5 | 3 | 3 |
| 12 | 2 | 2 | 1 |
| Сума балів | $СБ_1$=31 | $СБ_2$=34 | $СБ_3$=34 |
| Середньоарифметична сума балів $\overline{СБ}$ | $\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{31+34+34}{3} = 33$ | | |

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 33 бали, що згідно таблиці 5.2 вважається, що рівень комерційного потенціалу проведених досліджень є вище середнього.

Реалізація проекту можлива у навчальних закладах, зменшивши витрати на різні аналогічні системи тестування, корисно для віддаленого навчання так як простий інтерфейс дає змогу легко розібратись з усіма функціями. Також реалізувати дану розробку можливо в комерційних фірмах де відбувається контроль знань працівників адже тестування допоможе визначити ріст працівника протягом усього часу роботи та відповідно коригування заробітної плати та витрат

Порівняємо нову розробку з аналогами, які існують на ринку. В якості аналога для розробки було обрано Kahoot, ISpring, QuizMaker. Основними недоліками аналога є: ISpring – сервіс не може бути використано для первырки знань учнів, Kahoot не можливо автоматизувати оцінку знань працівників в компанії, QuizMaker – не портативний, важко в налаштування то розгортці програмного продукту. Також до недоліків можна віднести відсутність автоматичного коригування системи складності рівня запитань.

У розробці дана проблема вирішується за допомогою розробки власного додатку - з простою розгорткою, універсальністю використання, дешевизною. Також система випереджає аналог за такими параметрами як системою автоматичного коригування рівня складності.

## 5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

1. Основна заробітна плата кожного із дослідників $З_о$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)} \quad (4.1)$$

де $M$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$T_P$ – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

$t$ – число робочих днів роботи дослідника.

Для розробки програмного засобу діагностики знань з автоматичним коригуванням рівня складності запитань необхідно залучити програміста з посадовим окладом 8000 грн. Кількість робочих днів у місяці складає 22, а кількість робочих днів програміста складає 22. Зведемо сумарні розрахунки до таблиця 5.4.

Таблиця 5.4 – Заробітна плата дослідника в науковій установі бюджетної сфери

| Найменування посади | Місячний посадовий оклад, грн. | Оплата за робочий день, грн. | Число днів роботи | Витрати на заробітну плату грн. |
|---|---|---|---|---|
| Керівник | 12000 | 545,5 | 5 | 2727 |
| Програміст | 8000 | 363,6 | 22 | 8000 |
| Тестувальник | 7500 | 340,9 | 10 | 3409 |
| Всього | | | | 14136 |

2. Розрахунок додаткової заробітної плати робітників

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{доп}}{100\%} \quad (4.2)$$

$$З_д = 0{,}11 * 14136 = 1555 \text{ (грн)}$$

3. Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (4.3):

$$Н_{ЗП} = (З_о + З_д) * \frac{\beta}{100} \text{ (грн)} \quad (4.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

$\beta$ – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді**:**

$$Н_{ЗП} = (14136 + 1555) * \frac{22}{100} = 3452{,}1 \text{ (грн)}$$

4. Витрати на матеріали М та комлектуючі К, що були використані під час виконання даного етапу роботи, розраховуються по кожному виду матеріалів за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ грн.,} \quad (4.4)$$

де $Н_i$ – витрати матеріалу *i*-го найменування, кг;

$Ц_i$ – вартість матеріалу *i*-го найменування, грн./кг.;

$К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15);

$В_i$ – маса відходів матеріалу *i*-го найменування, кг;

$Ц_в$ – ціна відходів матеріалу *i*-го найменування, грн/кг;

n – кількість видів матеріалів.

Таблиця 5.5 – Матеріали, що використані на розробку

| Найменування матеріалу | Ціна за одиницю, грн. | Витрачено | Вартість витраченого матеріалу, грн. |
|---|---|---|---|
| Папір | 130 | 1 | 130 |
| Ручка | 15 | 2 | 30 |
| CD-диск | 18 | 1 | 18 |
| Флешка | 200 | 1 | 200 |
| Всього | | | 378 |
| З врахуванням коефіцієнта транспортування | | | 415,8 |

5. Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} \cdot С_{прг.і} \cdot К_{і,} \quad (4.5)$$

де $Ц_{іпрг}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($К_і = 1,10...1,12$);

$k$ – кількість найменувань програмних засобів.

Таблиця 5.6 – Витрати на придбання програмних засобів по кожному виду

| Найменування устаткування | Кількість, шт | Ціна за одиницю, грн | Вартість |
|---|---|---|---|
| Itelij Idea | 1 | 3000 | 3000 |
| Сервер | 1 | 10000 | 10000 |
| Всього | | | 14300 |

6. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$A = \frac{Ц \cdot T}{Ткор \cdot 12} \quad [грн], \tag{4.6}$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункта 137.3.3 Податкового кодекса амортизація нараховується на основні засоби вартістю понад 2500 грн. В нашому випадку для написання магістерської роботи використовувався персональний комп'ютер вартістю 17000 грн.

$$А = \frac{17000 \cdot 1}{2 \cdot 12} = 708{,}33$$

7. До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i} \tag{4.7}$$

де $W_{yt}$ – встановлена потужність обладнання на певному етапі розробки, кВт;

$t_i$ – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

$\eta_i$ – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}3 \cdot 185 \cdot 4{,}1 \cdot 0{,}5}{0{,}8} = 142{,}22$$

Витрати на службові відрядження, витрати на роботи, які виконують сторонні підприємства, установи, організації та інші витрати в нашому досліджені не враховуються оскільки їх не було.

Накладні (загальновиробничі) витрати $В_{нзв}$ охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 14136 \cdot \frac{100}{100\%} = 14136 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

$$В = 14136+1555+3452,1+415,8+14300+708,33+142,22+14136=48846,2$$

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (4.9)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт $\beta = 0,9$.

Звідси:

$$ЗВ = \frac{48846,2}{0,9} = 54273,53 \text{ грн.}$$

## 5.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_{1}^{n} \left( \Delta Ц_o \cdot N + Ц_o \cdot \Delta N \right)_i \cdot \lambda \cdot \rho \cdot \left( 1 - \frac{\upsilon}{100} \right) \qquad (4.10)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

л – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х – ставка податку на прибуток. У 2021 році – 18%.

Припустимо, що при впроваджені результатів наукової розробки покращується якість програмного продукту для формування індивідуальних тренувань. Припустимо, що ціна від зросте на 500 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 200 шт.,

протягом другого року – на 300 шт., протягом третього року на 400 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до складає 1000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [500 \cdot 1 + (1000 + 500) \cdot 200] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 51333{,}36 \text{ грн.}$$

$$\Delta П_2 = [500 \cdot 1 + (1000 + 500) \cdot (200 + 300)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 128619{,}88 \text{ грн.}$$

$$\Delta П_3 = [500 \cdot 1 + (1000 + 500) \cdot (200 + 300 + 400)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 231115{,}78 \text{ грн.}$$

## 5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \qquad (4.11)$$

$k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}} = 2…5$).

$$PV = 2 \cdot 54273{,}53 = 108547{,}06$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{абс} = (ПП - PV) \tag{4.12}$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{4.13}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

$\tau$ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$\text{ПП} = \frac{51333{,}36}{(1+0{,}2)^1} + \frac{128619{,}88}{(1+0{,}2)^2} + \frac{231115{,}78}{(1+0{,}2)^3} = 266466{,}79 \text{ грн.}$$

$$\text{Е}_{\text{абс}} = (266466{,}79 - 108547{,}06\ ) = 157919{,}73 \text{грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.14}$$

$T_{ж}$ – життєвий цикл наукової розробки, роки.

$$\mathrm{E_в} = \sqrt[3]{1 + \frac{157919{,}73}{108547{,}06}} - 1 = 0{,}58 = 58\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f,$$

(4.15)

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d= (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{\min} = 0{,}18 + 0{,}05 = 0{,}23.$$

Так як $E_в > \tau_{\min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} \qquad (4.16)$$

$$\mathrm{T_{ок}} = \frac{1}{0{,}58} = 1{,}7 \text{ роки}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

## 5.5 Висновки до економічного розділу

Було проведено оцінку комерційного потенціалу методу та програмного засобу діагностики знань з автоматичним коригуванням рівня складності запитань, який є на вище середньому рівні.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складе 48846,2 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 54273,53 грн.

Вкладені інвестиції в даний проект окупляться через 1,7 роки при прогнозованому прибутку 266466,79 грн. за три роки.

# ВИСНОВКИ

У магістерській кваліфікаційній роботі було розроблено додаток для створення індивідуальної системи тестування з автоматичним коригування рівня складності запитань, дана розробка складається з двох основних частин: інтерфейсу та серверної частини додатку що призначена для обробка даних. частина створена в середовищі розробки Visual Studio на мові C#, а серверна частина написана на мові Java в середовищі розробки Inteliij Idea.

Було проаналізовано стан питання, актуальність завдання, а також методи його вирішення. Визначено основі аналоги в даній сфері та проведено їх аналіз для покращення та виправлення їх недоліків у своєму проекті.

Обрано мови програмування C#, Java та середовища програмування Visual Studio та Inteliij Idea для реалізації поставленої задачі.

Були вирішенні такі задачі:

- проаналізовано існуючі системи і засоби тестування
- виявлено основні недоліки в існуючих аналогів та врахувати їх для створення власного додатку
- розроблено основний алгоритм системи тестування;
- розроблено сучасний дизайн, що дасть змогу орієнтуватись в ньому без додаткового навчання;
- модифіковано метод автоматичного коригування рівня складності запитань для створення системи тестувань;
- створено просту універсальну платформу тестування;
- розроблено алгоритм автоматичного коригування рівня складності запитань
- проведено тестування програмного продукту для перевірки всіх можливих варіантів використання.

При реалізації програми та створення графічного інтерфейсу користувача було обрано інструмент програмування додатків Windows Forms, для створення серверної частини була застосована технологія Spring.

Перед написанням програмного продукту було розроблено схеми алгоритмів тестування, а також діаграму усіх базових компонентів.

Тестування працездатності програмного додатку було виконано у повній мірі, перевірено коректність обмеження на ввід даних, надсилання запитів на сервер та отримання даних користувачем, а також якість даних що надсилаються та зберігаються в базі, що продемонструвало відповідність додатку до усіх поставлених завдань.

Виконано економічний аналіз та розрахунки, результати яких підтвердили доцільність розробки.

# СПИСОК ВИКОРИСТАНХ ДЖЕРЕЛ


1. Стандартизоване тестування. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Стандартизоване_тестування.
2. Інформаційні технології та технічні засоби навчання. [Електронний ресурс] – Режим доступу: https://pidruchniki.com/12590605/informatika/formi_testovih_zavdan_forma_podannya_testovogo_zavdannya
3. Інформаційні технології та технічні засоби навчання. [Електронний ресурс] – Режим доступу: https://pidruchniki.com/10611207/informatika/vikoristannya_testovih_sistem
4. Kahoot! – онлайн-сервіс для створення вікторин. [Електронний ресурс] – Режим доступу: https://www.pedrada.com.ua/news/276-kahoot-onlajjn-servis-dlja-stvorennja-viktorin-didaktichnikh-igor-i-testiv
5. Kahoot [Електронний ресурс] – Режим доступу: https://kahoot.it
6. ISpring [Електронний ресурс] – Режим доступу: https://www.ispring.ru/ispring-quizmaker
7. QuizMaker [Електронний ресурс] – Режим доступу: https://www.quiz-maker.com/
8. Руководство по программированию в Windows Forms. [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/windowsforms/
9. PostgreSQL. [Електронний ресурс] – Режим доступу: https://habr.com/ru/hub/postgresql/
10. Spring Framework. [Електронний ресурс] – Режим доступу: https://ru.wikibooks.org/wiki/Spring_Framework
11. Windows Forms. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Windows_Forms
12. Windows Forms - Windows Forms. [Електронний ресурс] – Режим доступу: https://ru.qwe.wiki/wiki/Windows_Forms
13. Еккель Брюс. Філософія Java. - М.: Бомбора (Ексмо), 2019. – 1168 с.

14. Ruby - Windows Forms. [Електронний ресурс] – Режим доступу: https://ru.wikibooks.org/wiki/Ruby
15. C++ - Windows Forms. [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/C%2B%2B
16. Java - Windows Forms. [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Java
17. Патрик Неймер, Програмування на Java. – М.: Ексмо, 2017. – 1214 с.
18. Гербер Шилдт. Java. Руководство для початківців – М.: Диалектика-Вильямс, 2012. – 624 с.
19. Гербер Шилдт. Java 8 для початківців – М.: Диалектика-Вильямс, 2015. – 720 с.
20. Сереалізація на мові Java [Електронний ресурс] – Режим доступу: URL: https://habr.com/ru/post/60317/. – Назва з екрану.
21. Посібник Java [Електронний ресурс]. – Режим доступу: https://code.makery.ch/ru/library/javafx-tutorial/part1/ – Назва з екрану.
22. Spring Framework [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Spring_Framework
23. Обзор Windows Forms [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/framework/winforms/windows-forms-overview
24. Авторизація. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Авторизація
25. Формат JSON, метод toJSON. [Електронний ресурс] – Режим доступу: https://learn.javascript.ru/json
26. JSON. [Електронний ресурс] – Режим доступу: https://gist.github.com/ermakovpetr/4c9f56d48e49d822705a
27. PostgreSQL. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/PostgreSQL

28. Що таке тестування програмного забезпечення. [Електронний ресурс] – Режим доступу: https://www.quality-assurance-group.com/shho-take-testuvannya-programnogo-zabezpechennya-ta-yake-jogo-znachennya/
29. Тестування програмно продукту. [Електронний ресурс] – Режим доступу: http://lib.mdpu.org.ua/e-book/vstup/L11.html
30. Postman. [Електронний ресурс] – Режим доступу: https://software-testing.ru/library/testing/testing-tools/2638-postman
31. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

# Додатки

Додаток А – Технічне завдання

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"01" жовтня 2020 року

# ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи
«Розробка методу та програмного засобу діагностики знань з автоматичним коригуванням рівня складності запитань»

Науковий керівник: к.т.н., доц. каф. ПЗ
_____ Бабюк Н.П.
Магістрант групи 1ПІ-20м
_____ Струбчевський А.Г

Вінниця 2021

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність даного дослідження визначається необхідністю вирішення основних проблем існуючих засобів створення індивідуальних систем тестування з автоматичним коригуванням рівня складності запитань для точного та якісного контролю знань.

1.2 Наказ про затвердження теми магістерської кваліфікаційної роботи.

2 Мета і призначення МКР

2.1 Мета магістерської роботи полягає у підвищенні ефективності та якості проведення тестування.

2.2 Призначення розробки — виконання магістерської кваліфікаційної роботи.

3 Вихідні дані для виконання МКР

Виконати розробку програмного забезпечення для створення індивідуальної системи тестування з автоматичним коригуваннням рівня складності запитань, усі лістинги та алгоритми надати у додатках до роботи.

4 Вимоги до виконання МКР

МКР повинна задовольняти такі вимоги:

— запропонувати покращення точності методом електронного тестування рівнянь знань;

— розробити алгоритм для автоматичного коригування рівня складності запитань;

— вхідні дані — текстовий набір питань у відповідності до формату питання — відповідь;

— результат роботи, набір тестових запитань та користувачів в базі даних.

5 Етапи МКР та очікувані результати таблиця А.1

Таблиця А.1 — Етапи виконання роботи

| № | Назва етапу | Термін виконання | | Очікувані результати |
|---|---|---|---|---|
| | | початок | кінець | |
| 1 | Аналіз завдання. Вступ | 14.09.21 | 16.09.21 | Вступ |
| 2 | Аналіз літературних джерел для розпізнавання особи | 17.09.21 | 22.09.21 | розділ 1 |
| 3 | Розробка технічного завдання | 22.09.2021 | 25.09.21 | Технічне завдання |
| 3 | Розробка алгоритму та системи тестування | 25.09.21 | 13.10.21 | Розділ 2, розробка структури |
| 4 | Розробка елементів додатку | 11.10.21 | 29.10.21 | Розділ 3, розробка програми |
| 5 | Практична реалізація | 01.11.21 | 14.11.21 | Розділ 3 |
| 6 | Проведення тестування | 15.11.21 | 30.11.21 | Розділ 4 |
| 7 | Розрахунок економічної частини | 30.11.21 | 05.12.21 | Розділ 5 |
| 8 | Оформлення пояснювальної записки | 05.12.21 | 15.12.21 | ПЗ, презентація |

6 Матеріали, що подаються до захисту МКР: пояснювальна записка МКР, ілюстративні та графічні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив рецензента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів розрахункової та графічної документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР

Вимоги викладені в «Положенні про порядок підготовки магістрів у Вінницькому національному технічному університеті» з урахуванням змін, що подані у бюлетені ВАК України № 9-10, 2011р., а також в МЕТОДИЧНИХ

ВКАЗІВКАХ до дипломного проектування, ДСТУ 3008-2015, ДСТУ 3974-2000 «Правила виконання дослідно-конструкторських робіт. Загальні положення» та діючого ГОСТ 2.114-95 ЄСКД.

9 Вимоги щодо технічного захисту інформації в МКР з обмеженим доступом відсутні.

Технічне завдання до виконання отримав _____ Струбчевський А. Г.

# Додаток Б

Додаток В – Лістинг програми

```
using EasyQuiz.Quiz;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace EasyQuiz.Authorization
{
   [DataContract]
   [Serializable]
  public class NewAdmin
   {
      [DataMember]
      string name;
      [DataMember]
      string surname;
      [DataMember]
      string email;
      [DataMember]
      string login;
      [DataMember]
      string password;
      [DataMember]
      List<Quizes> quizzes;
      [DataMember]
      bool isadmin;

      public NewAdmin()
      {
      }

      public NewAdmin(string name, string surname, string email, string login, string
password, List<Quizes> quizzes, bool isadmin)
      {
         this.name = name;
         this.surname = surname;
         this.email = email;
         this.login = login;
         this.password = password;
```

```
            this.quizzes = quizzes;
            this.isadmin = isadmin;
        }

        public void setNewUser(string name, string surname, string email, string login,
string password, bool isadmin)
        {
            this.name = name;
            this.surname = surname;
            this.email = email;
            this.login = login;
            this.password = password;
            this.isadmin = isadmin;
            Console.WriteLine(name, surname, email, login, password, isadmin);


        }


    }
}

namespace EasyQuiz.Authorization
{
    class Role
    {
        private string roleName;
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;

namespace EasyQuiz.Quiz
{
    public class Quizes
    {
        string question;
        string answer1;
        string answer2;
```

```
        string answer3;
        string answer4;
    }
}


using EasyQuiz.Authorization;
using System.Text.Json;
using System.Text.Json.Serialization;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.Serialization.Json;
using Newtonsoft.Json;
using System.Net.Http.Headers;
using EasyQuiz.Quiz;
using System.Net;
using System.IO;

namespace EasyQuiz.IOInfo
{
    class IOHttpTransfer
    {
        public bool isexist;
        private static readonly HttpClient client = new HttpClient();


        public void DoPostAdmin(NewAdmin jsonContent, HttpMethod method, string
urlPath) {
            DataContractJsonSerializer formatter = new
DataContractJsonSerializer(typeof(NewAdmin));

            string serialized = JsonConvert.SerializeObject(jsonContent);
            var json = JsonConvert.SerializeObject(jsonContent);

            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri("http://localhost:8080");

            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));//ACCEPT header
```

```
            HttpRequestMessage request = new HttpRequestMessage(method, urlPath);
            request.Content = new StringContent(json, Encoding.UTF8,
"application/json");
            client.SendAsync(request).ContinueWith(responseTask =>
            {
               if (!responseTask.Result.IsSuccessStatusCode) {
                  throw new ArgumentException();

               };


            });
         }
         public String DoGetIdAdmin(HttpMethod method, string urlPath)
         {
            DataContractJsonSerializer formatter = new
DataContractJsonSerializer(typeof(NewAdmin));

            WebRequest request = WebRequest.Create(urlPath);

            using (HttpWebResponse response =
(HttpWebResponse)request.GetResponse())
            using (Stream stream = response.GetResponseStream())
            using (StreamReader reader = new StreamReader(stream))
            {
               return reader.ReadToEnd();
            }

            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri("http://localhost:8080");

            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));//ACCEPT header

            //HttpRequestMessage request = new HttpRequestMessage(method, urlPath);
            //request.Content = new StringContent(Encoding.UTF8, "application/json");

            client.GetAsync(urlPath);
         }
         public void DoPostQuizes(Quizes jsonContent, HttpMethod method, string
urlPath)
         {
```

```
        DataContractJsonSerializer formatter = new
DataContractJsonSerializer(typeof(Quizes));

        string serialized = JsonConvert.SerializeObject(jsonContent);
        var json = JsonConvert.SerializeObject(jsonContent);

        HttpClient client = new HttpClient();
        client.BaseAddress = new Uri("http://localhost:8080");

        client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));//ACCEPT header

        HttpRequestMessage request = new HttpRequestMessage(method, urlPath);
        request.Content = new StringContent(json, Encoding.UTF8,
"application/json");
        client.SendAsync(request).ContinueWith(responseTask =>
        {
            if (!responseTask.Result.IsSuccessStatusCode)
            {
                throw new ArgumentException();
            };

        });
    }


  }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EasyQuiz
{
  public partial class Form1 : Form
  {
    public Form1()
```

```
{
   InitializeComponent();
   customizeDesign();
}

private void metroButton2_Click(object sender, EventArgs e)
{
          if (Login.Visible == true) {
      Login.Visible = false;
   }
   else if (Login.Visible == false) {
      Login.Visible = true;
   }
}
private void button1_Click(object sender, EventArgs e)
private void metroTrackBar1_Scroll(object sender, ScrollEventArgs e)

private void Form1_Load(object sender, EventArgs e)

private void panel1_Paint(object sender, PaintEventArgs e)
{

}
private Form activeForm = null;
private void openChildForm(Form childForm)
{
   if (activeForm != null) activeForm.Close();
   activeForm = childForm;
   childForm.TopLevel = false;
   childForm.FormBorderStyle = FormBorderStyle.None;
   childForm.Dock = DockStyle.Fill;
   panelChildForm.Controls.Add(childForm);
   panelChildForm.Tag = childForm;
   childForm.BringToFront();
   childForm.Show();
}
private void Login_Click_1(object sender, EventArgs e)
{
   openChildForm(new Login());
   hideSubMenu();
}
private void customizeDesign() {
   panelProfileSubmenu.Visible = false;
   panelProfSubMenu.Visible = false;
```

```
    panelSubSettings.Visible = false;
}
private void hideSubMenu() {
    if (panelProfileSubmenu.Visible == true) {
        panelProfileSubmenu.Visible = false;
    }
    if (panelProfSubMenu.Visible == true)
    {
        panelProfSubMenu.Visible = false;
    }
    if (panelSubSettings.Visible == true)
    {
        panelSubSettings.Visible = false;
    }
}
private void showSubMenu(Panel subMenu)
{
    if (subMenu.Visible == false)
    {
        subMenu.Visible = true;
    }
    else
        subMenu.Visible = false;
}
private void Profile_Click(object sender, EventArgs e)
{
    showSubMenu(panelProfileSubmenu);

private void button2_Click(object sender, EventArgs e)
{
    openChildForm(new Registration());
    hideSubMenu();
}

private void SaveFile_Click(object sender, EventArgs e)
{
    hideSubMenu();
}

private void DownloadFile_Click(object sender, EventArgs e)
{
    hideSubMenu();
private void AboutProgram_Click(object sender, EventArgs e)
{
```

```
            hideSubMenu();
        }
        private void Languages_Click(object sender, EventArgs e)
        {
            hideSubMenu();
        }
        private void File_Click(object sender, EventArgs e)
        {
            showSubMenu(panelProfSubMenu);
        }
        private void Settings_Click(object sender, EventArgs e)
        {
            showSubMenu(panelSubSettings);
        }
    }
}
package com.easyquiz.EasyQuizServer.controller;

import com.easyquiz.EasyQuizServer.entity.User;
import com.easyquiz.EasyQuizServer.service.UserService;
import lombok.AllArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * Created by sgva1 on 4/5/2020.
 */

@RestController
@RequestMapping("/user")
@AllArgsConstructor

public class UserController {

    private UserService userService;

    public void UserController(){}
    @PostMapping
    public ResponseEntity registerUser(@RequestBody String json) {
        return ResponseEntity.ok(userService.save(new User()));
```

```
    }
}

package com.easyquiz.EasyQuizServer.entity;

import com.easyquiz.EasyQuizServer.controller.UserController;
import lombok.Data;

import javax.persistence.*;

/**
 * Created by sgva1 on 4/5/2020.
 */

@Table(name = "answers")
@Entity
@Data
public class Answer {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
//    private String answer;
//    @ManyToOne
//    private Quize quize;
}


package com.easyquiz.EasyQuizServer.entity;

import lombok.Data;

import javax.persistence.*;
import java.util.Set;

/**
 * Created by sgva1 on 4/5/2020.
 */

@Data
@Entity
@Table(name = "quize")
public class Quize {

    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
//    private String question;
//    @OneToMany
//    private Set<Answer> answers;
}
```

```
package com.easyquiz.EasyQuizServer.entity;

import lombok.Data;

import javax.persistence.*;
import java.util.List;

/**
 * Created by sgva1 on 4/5/2020.
 */

@Data
@Entity
@Table(name = "quize_users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
//    private String name;
//    private String surname;
//    private String username;
//    private String password;
//    private String email;
//    private List<Quize> quizes;
//    private boolean isAdmin;
}
```

```
package com.easyquiz.EasyQuizServer.repository;

import com.easyquiz.EasyQuizServer.entity.User;
import org.springframework.data.jpa.repository.JpaRepository;

/**
 * Created by sgva1 on 4/5/2020.
 */
```

```
public interface UserRepository extends JpaRepository<User, Long> {
}

package com.easyquiz.EasyQuizServer.service;

import java.util.Collection;

/**
 * Created by sgva1 on 4/5/2020.
 */
public interface BaseMethods<T> {
    T save(T t);
    T getById(Long id);
    Collection<T> getAll();
    T edit(T t);
    void delete(T t);
}

package com.easyquiz.EasyQuizServer.service;

import com.easyquiz.EasyQuizServer.entity.User;

/**
 * Created by sgva1 on 4/5/2020.
 */
public interface UserService extends BaseMethods<User> {
}

package com.easyquiz.EasyQuizServer.service;

import com.easyquiz.EasyQuizServer.entity.User;
import com.easyquiz.EasyQuizServer.repository.UserRepository;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.Collection;

/**
 * Created by sgva1 on 4/5/2020.
 */

@Service
@AllArgsConstructor
public class UserServiceImpl implements UserService {
```

```
    private UserRepository userRepository;

    @Override
    public User save(User user) {
        return userRepository.save(user);
    }

    @Override
    public User getById(Long id) {
        return userRepository.getOne(id);
    }

    @Override
    public Collection<User> getAll() {
        return userRepository.findAll();
    }

    @Override
    public User edit(User user) {
        return userRepository.saveAndFlush(user);
    }

    @Override
    public void delete(User user) {
        userRepository.delete(user);
    }
}
```

```
package com.easyquiz.EasyQuizServer;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class EasyQuizServerApplication {

   public static void main(String[] args) {
      SpringApplication.run(EasyQuizServerApplication.class, args);
   }

}
```

```
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EasyQuiz
{

    public partial class CreateQuiz : Form
    {
        //public Form MainForm { get; set; }
        //static bool createQuiz = false;
        MyDelegate d;
        public CreateQuiz(MyDelegate sender)
        {
            InitializeComponent();
            d = sender;
        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void pictureBox4_Click(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
```

```
        {
            //Form1 main = this.Owner as Form1;
            //QuizGenerator quizGenerator = new QuizGenerator();
            d(Convert.ToInt32(textBox1.Text), textBox2.Text);

            // quizGenerator.MdiParent = MainForm;


            //quizGenerator.Show();
            //Form1 form1 = new Form1();
            //form1.openChildForm(new QuizGenerator());
            this.Close();

        }

        public void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void CreateQuiz_Load(object sender, EventArgs e)
        {

        }

        private void metroCheckBox1_CheckedChanged(object sender, EventArgs e)
        {

        }
    }
}

using EasyQuiz.Quiz;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EasyQuiz
```

```
{
   public partial class Testing : Form
   {
      public int mark;
      public int Amounte;
      int id; string name; string question; string answer1; string answer2; string
answer3; string answer4; public int right_answer;
      List<Quizes> listfirst1;
      int diap1;
      int diap2;
      double right_answers;
      double amount_answers;
      double percentage;
      Boolean easyQuest = false;
      public Testing(HashSet<Quizes> quizes, int dia1, int dia2)
      {

         diap1 = dia1;
         diap2 = dia2;

         InitializeComponent();
         List<Quizes> listfirst = new List<Quizes>(quizes);
         listfirst1 = listfirst;
         herewego(listfirst, dia1);
         comboBox1.Items.Add(1);
         comboBox1.Items.Add(2);
         comboBox1.Items.Add(3);
         comboBox1.Items.Add(4);
         foreach (Quizes firstLevel in quizes)
         {

            Amounte++;


            // HashSet<Answer> _usersIDs = new
HashSet<Answer>(firstlevel.Select(user => user.ID));


            //foreach (Answer im in firstLevel.Answers)
            //{


            //    // im.
            //    //im.Equals(0);
```

```
            //}


            // Console.Write(" {0}", i);
        }
    }


    void herewego(List<Quizes> listfirst, int number) {


        if (easyQuest == true)

        {
            for (int i = number; i < listfirst.Capacity; i++) {
                if (listfirst[i].Hardness < 3)
                {

                    label5.Text = listfirst[i].Question;
                    List<Answer> listanswerfirst = new
List<Answer>(listfirst[i].Answers);
                    right_answer = listanswerfirst[0].Right_answer;
                    char[] charsToTrim = { '*', ' ', '\" };

                    label6.Text = Convert.ToString(listanswerfirst[0].AnswerSet);
                    label9.Text = Convert.ToString(listanswerfirst[1].AnswerSet);
                    label10.Text = Convert.ToString(listanswerfirst[2].AnswerSet);
                    label11.Text = Convert.ToString(listanswerfirst[3].AnswerSet);
                    easyQuest = false;
                    break;
                }


            }



        }
        else {
            label5.Text = listfirst[number].Question;
            List<Answer> listanswerfirst = new
List<Answer>(listfirst[number].Answers);
            right_answer = listanswerfirst[0].Right_answer;
            char[] charsToTrim = { '*', ' ', '\" };
```

```
            label6.Text = Convert.ToString(listanswerfirst[0].AnswerSet);
            label9.Text = Convert.ToString(listanswerfirst[1].AnswerSet);
            label10.Text = Convert.ToString(listanswerfirst[2].AnswerSet);
            label11.Text = Convert.ToString(listanswerfirst[3].AnswerSet);
        }


        //HashSet<Answer> secondlevel = new
HashSet<Answer>(firstLevel.Answers);
        //List<Answer> listsecond = new List<Answer>(secondlevel);


        //for (int i = 0; number > i; i++)
        //{

        //}

    }
    private void tableLayoutPanel1_Paint(object sender, PaintEventArgs e)
    {

    }

    private void label6_Click(object sender, EventArgs e)
    {


    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (comboBox1.SelectedItem.Equals(right_answer))
        {
            MessageBox.Show("Вірно!!", "Внимание", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
            if (diap1 < diap2)
            {
                diap1++;
                amount_answers++;
                right_answers++;
                herewego(listfirst1, diap1);

            }
```

```
            else {
                percentage = (right_answers / amount_answers) * 100;
                MessageBox.Show(Convert.ToString(percentage + "%"), "Внимание",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
            }

        }

        else {
            MessageBox.Show("Не вірно!!", "Внимание", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
            if (diap1 < diap2)
            {
                diap1++;
                amount_answers++;
                herewego(listfirst1, diap1);

            }
            if (diap1 < diap2 && (percentage = (right_answers / amount_answers) *
100)<40)
            {
                diap1++;
                amount_answers++;
                easyQuest = true;
                herewego(listfirst1, diap1);

            }
            else
            {
                percentage = (right_answers / amount_answers) * 100;
                MessageBox.Show(Convert.ToString(percentage+"%"), "Внимание",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
            }

        };
    }
  }
}

using EasyQuiz.IOInfo;
using EasyQuiz.Quiz;
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EasyQuiz
{
    public partial class QuizGenerator : Form
    {
        Quizes quizes = new Quizes();
        int iduser;
        public int amounttests;
        QuizAmounte d;
        public QuizGenerator( int iduser, int amounttests, QuizAmounte sender)
        {
            InitializeComponent();

            comboBox1.Items.Clear();

            comboBox1.Items.Add(1);

            comboBox1.Items.Add(2);

            comboBox1.Items.Add(3);

            comboBox1.Items.Add(4);



            comboBox2.Items.Add(1);

            comboBox2.Items.Add(2);

            comboBox2.Items.Add(3);

            comboBox2.Items.Add(4);

            comboBox2.Items.Add(5);
            label9.Text = Convert.ToString(amounttests);
```

```
    this.amounttests = amounttests;
    this.iduser = iduser;
    d = sender;
}


private void label7_Click(object sender, EventArgs e)
{

}

private void tableLayoutPanel1_Paint(object sender, PaintEventArgs e)
{

}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
  //  comboBox1.Items.Add("sdsd");

}
void DisplaySet(HashSet<Quizes> collection)
{
    Console.Write("{");
    foreach (Quizes i in collection)
    {
        var im = i;

        Console.Write(" {0}", i);
    }
    Console.WriteLine(" }");
}
private void button1_Click(object sender, EventArgs e)
{

    //comboBox1.Items.Add("sdsd");
```

```
            quizes.Question = richTextBox1.Text;
            var rightAnswer = Convert.ToInt32(comboBox1.Text);
            quizes.User_id = iduser;
            quizes.Answers = new HashSet<Answer>();
            quizes.Answers.Add(new Answer(Convert.ToString(textBox1.Text),
rightAnswer));
            quizes.Answers.Add(new Answer(Convert.ToString(textBox2.Text),
rightAnswer));
            quizes.Answers.Add(new Answer(Convert.ToString(textBox3.Text),
rightAnswer));
            quizes.Answers.Add(new Answer(Convert.ToString(textBox4.Text),
rightAnswer));
            quizes.Hardness = Convert.ToInt32(comboBox2.Text);

            d(Convert.ToInt32(amounttests));

            new IOHttpTransfer().DoPostQuizes(quizes, HttpMethod.Post, "quize");
          // this.Close();
           richTextBox1.Text = "";
           textBox1.Text="";
           textBox2.Text="";
           textBox3.Text="";
           textBox4.Text="";
           amounttests--;
           label9.Text = Convert.ToString(amounttests);
           this.Refresh();
           if (amounttests == 0) {
              this.Close();
           }
}


      private void label5_Click(object sender, EventArgs e)
      {

      }

      private void QuizGenerator_Load(object sender, EventArgs e)
      {

      }

      private void label10_Click(object sender, EventArgs e)
      {
```

```
        }

        private void label9_Click(object sender, EventArgs e)
        {

        }

        private void label11_Click(object sender, EventArgs e)
        {

        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {

        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Runtime.Serialization;
using System.Text;
using System.Threading.Tasks;

namespace EasyQuiz.Quiz
{
    [DataContract]
    [Serializable]
   public class Quizes
    {
        [DataMember]
        string question;
        [DataMember]
        int user_id;
        [DataMember]
        private int hardness;

        [DataMember]
        private HashSet<Answer> answers;
```

```
        public string Question { get => question; set => question = value; }
        public int User_id { get => user_id; set => user_id = value; }
        public HashSet<Answer> Answers { get => answers; set => answers = value; }
        public int Hardness { get => hardness; set => hardness = value; }
    }
}
```

Додаток Г

**Ілюстративна частина**

РОЗРОБКА МЕТОДУ ТА ПРОГРАМНОГО ЗАСОБУ ДІАГНОСТИКИ ЗНАНЬ З АВТОМАТИЧНИМ КОРИГУВАННЯМ РІВНЯ СКЛАДНОСТІ ЗАПИТАНЬ

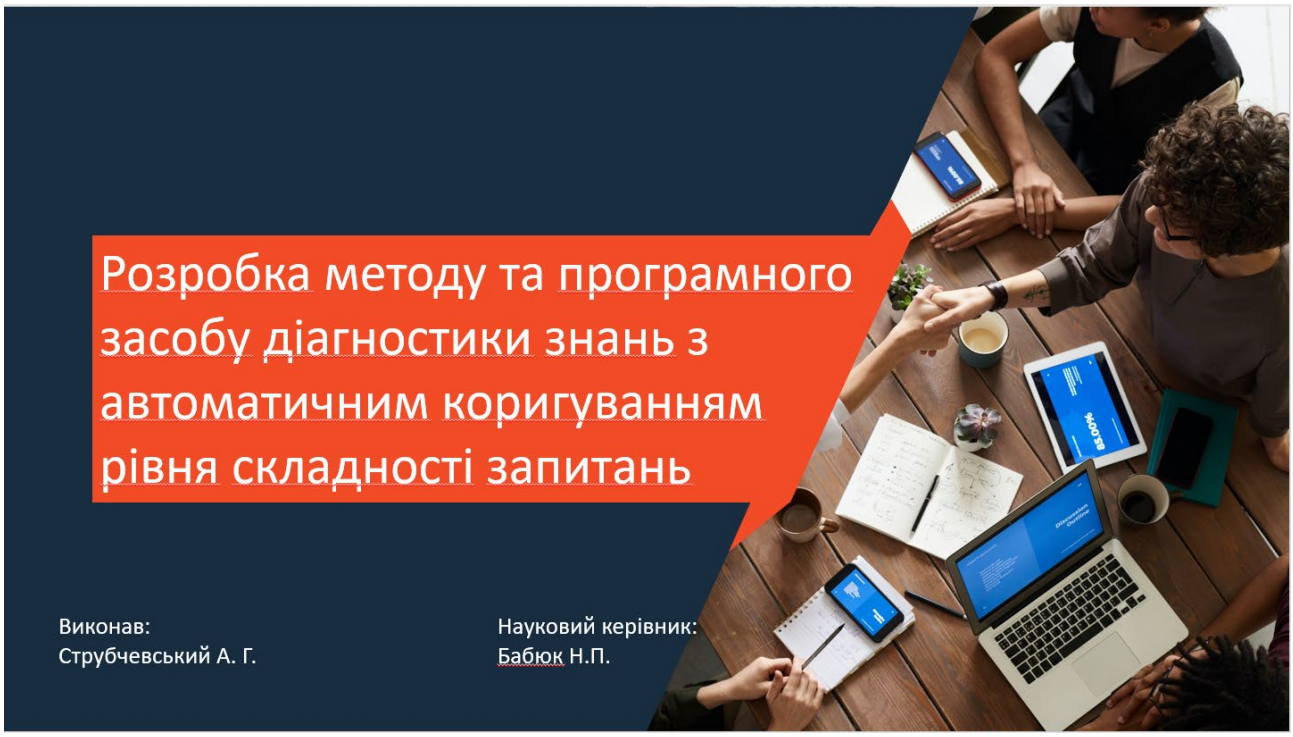



Рисунок Г.1 – Тема магістерської кваліфікаційної роботи

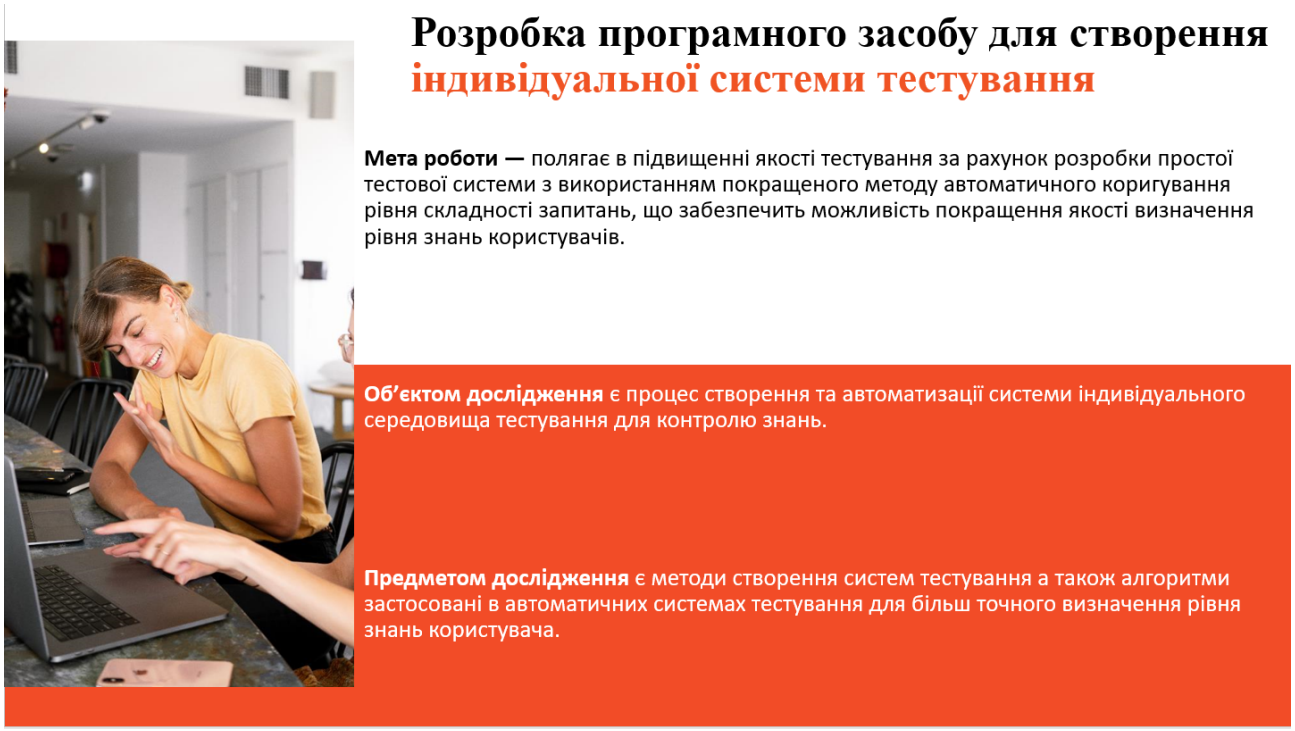



Рисунок Г.2 – Мета, завдання, об'єкт та предмет дослідження магістерської кваліфікаційної роботи

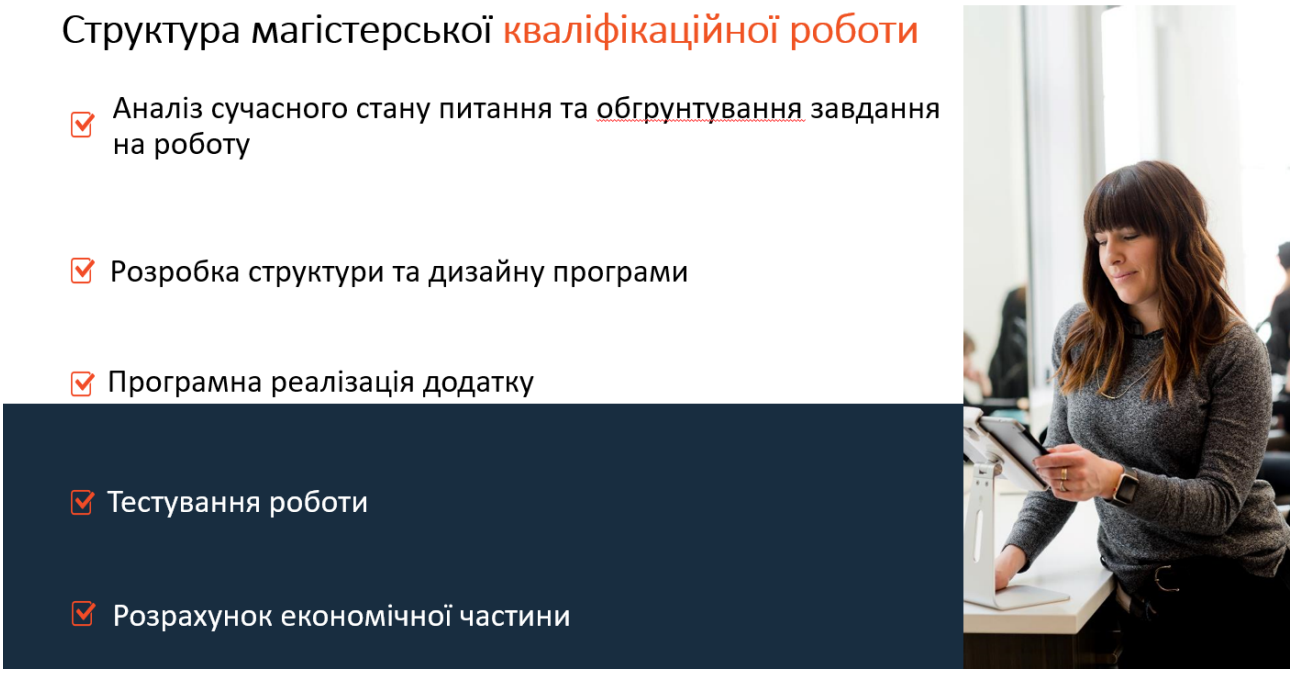



Рисунок Г.3 – Структура магістерської кваліфікаційної роботи

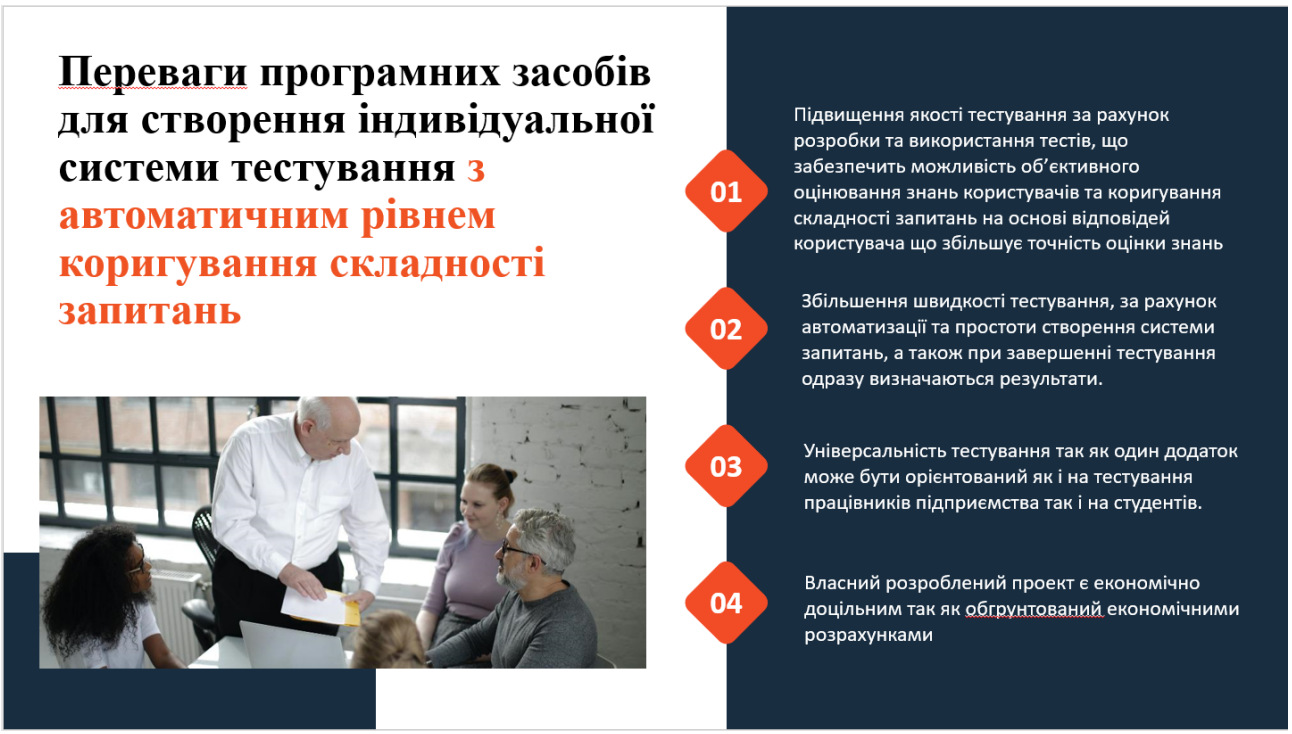



Рисунок Г.4 – Переваги програмних засобів для створення індивідуальної системи тестування


Рисунок Г.5 – Структура програмного додатку

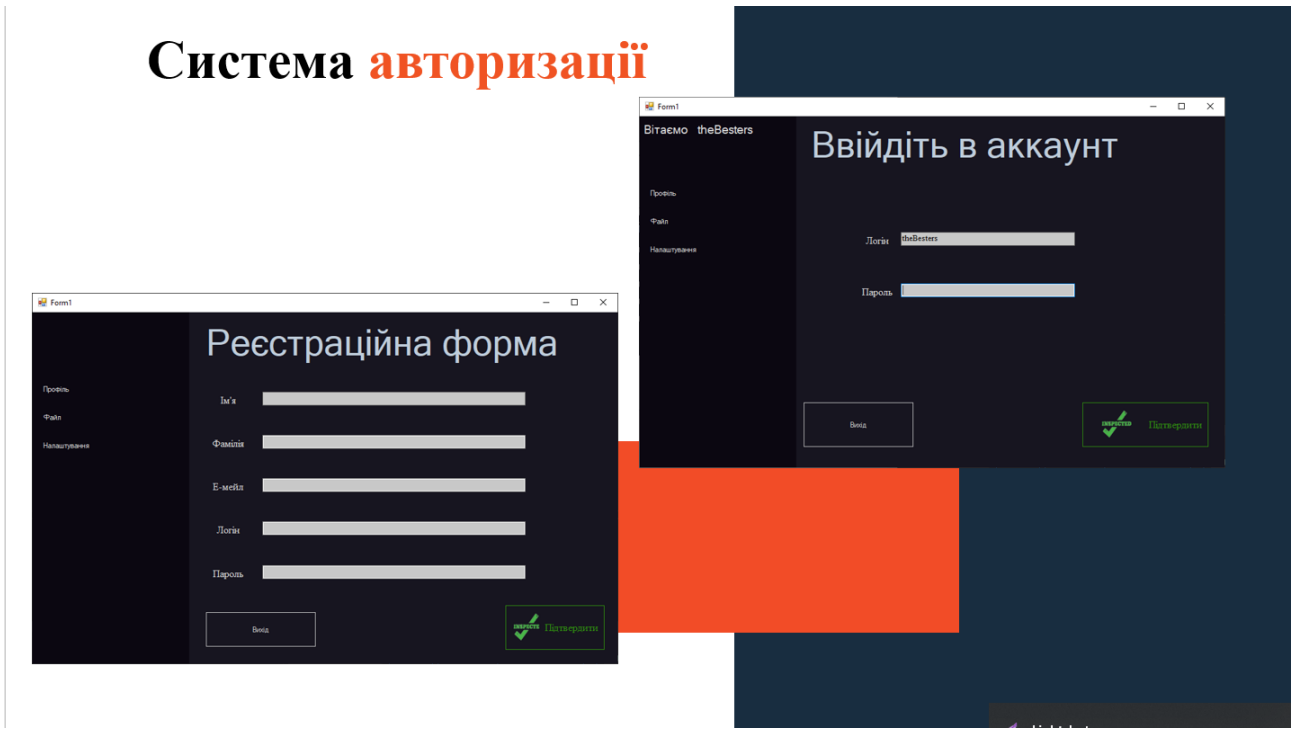



Рисунок Г.6 – Система авторизації програмного додатку

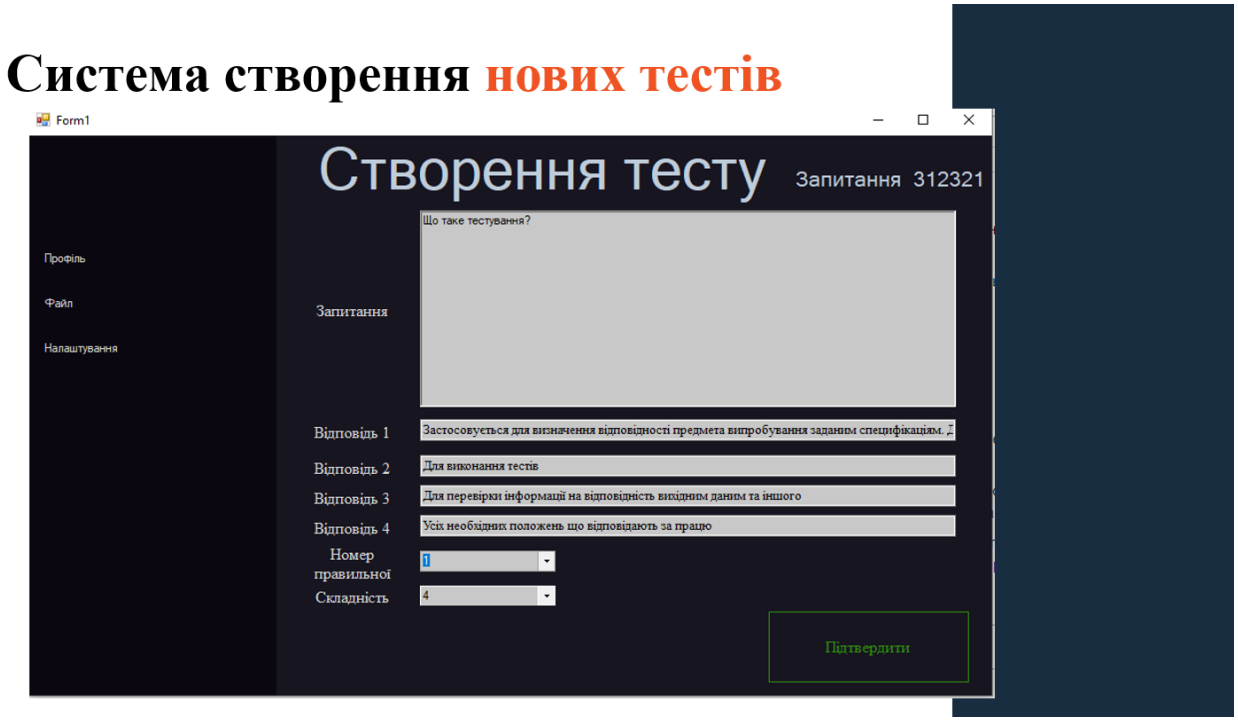



Рисунок Г.7 – Вікно створення нових тестів

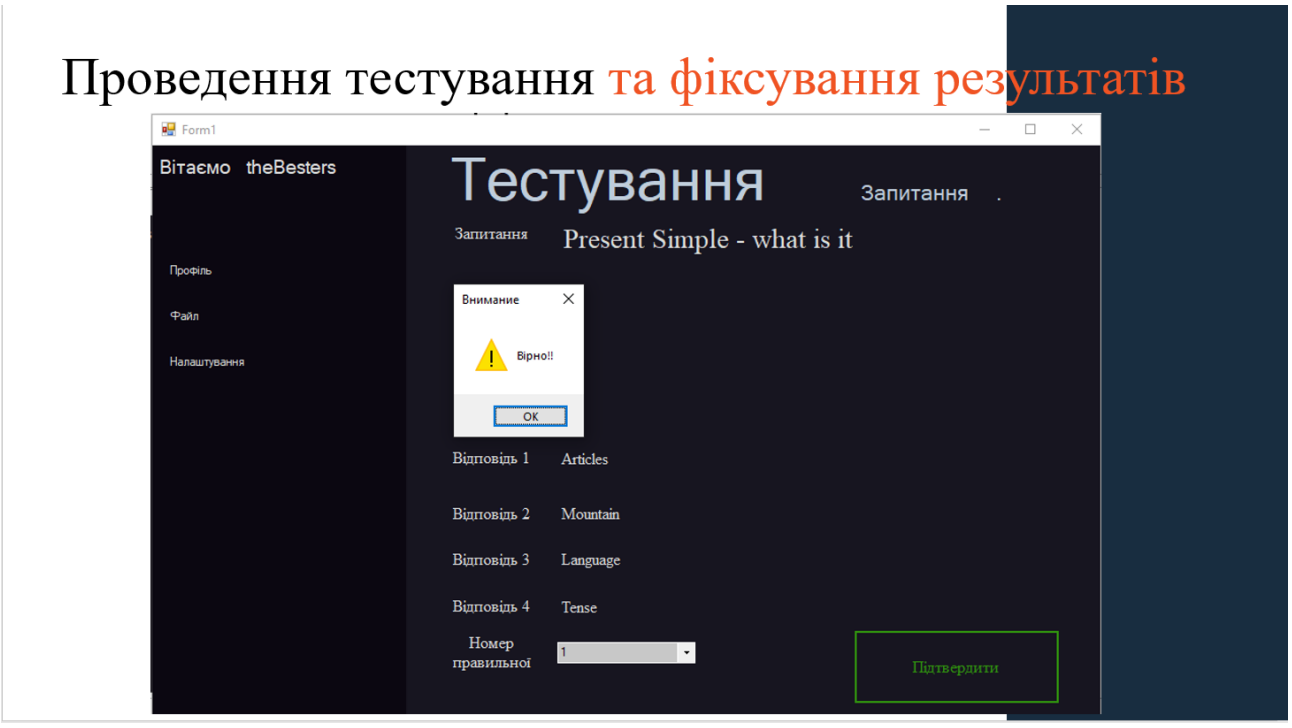



Рисунок Г.8 – Вікно проведення та фіксування результатів тестування

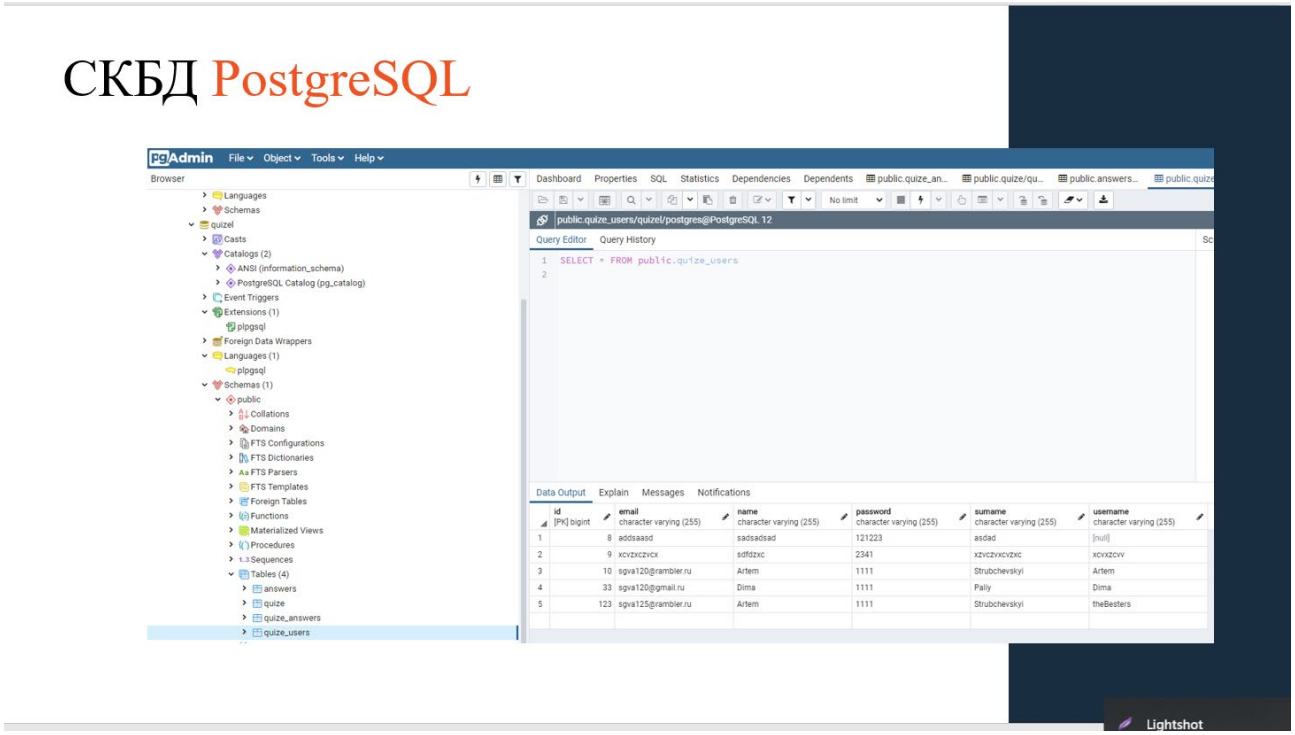



Рисунок Г.9 – СКБД PostgreSQL (PgAdmin)


Рисунок Г.10 – Алгоритм роботи системи коригування рівня складності запитань


Рисунок Г.11 – Висновки магістерської кваліфікаціної роботи

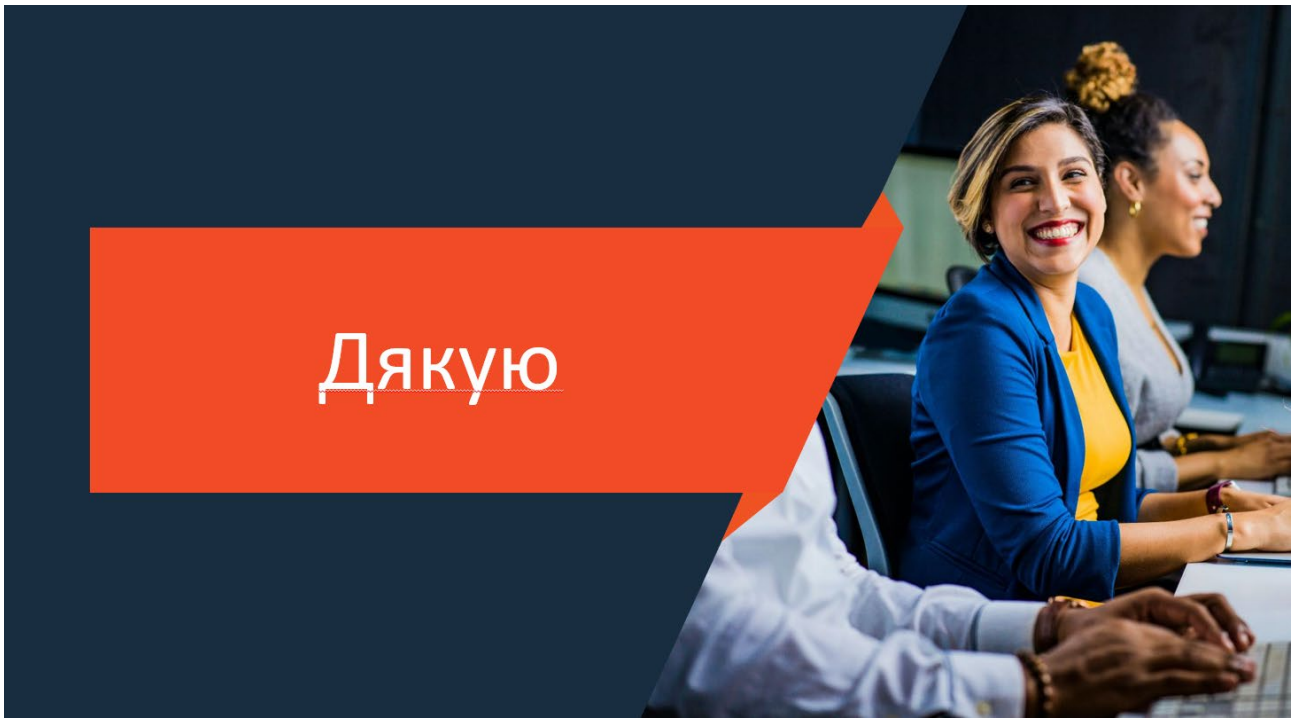



Рисунок Г.12 – Заключний слайд