**АНОТАЦІЯ**

Завданням бакалаврської кваліфікаційної роботи є розробка мобільного додатку для тестування знань з дисципліни «Організація комп’ютерних мереж», яка має допомагати студентам у підготовці до різнотипних іспитів, контрольних робіт тощо.

В процесі виконання завдання проведено аналіз предметної області, розглянуто методики для реалізації контролю знань та аналоги які частково їх реалізують. А також проведено аналіз інструментальних засобі за допомогою яким буде надана можливість коректної реалізації.

Для мобільного додатку для тестування знань розроблена специфікація вимог, що описує характеристику додатку в загальному, набір більш детальних характеристик системи, функціональні та нефункціональні вимоги для реалізації характеристик. Також реалізовано підсистему веб сервіс API, який повинен надавати можливість доступу до віддаленої бази даних.

На основі специфікації системи, розглянутих аналогів та предметної області загалом, здійснено проектування системи. Основою для реалізації архітектури системи було вибрано шаблон (патерн) проектування MVVM (Model View ViewModel).

Для написання мобільного додатку було вибрано мову програмування C# 6.0 та бібліотеку SQLite як СУБД для локальної бази даних. Для реалізації веб сервісу API використано Php Framework Symfony та бібліотеку API Platform. Для формування клієнтської частини для наповнення віддаленої бази даних використовується мова розмітки HTML та каскадні таблиці стилів CSS.

Виконання завдання є успішним та мобільний додаток з веб сервісом можуть бути застосовані для надання допомоги студентам у процесі підготовки до іспитів, контрольних робіт тощо, реалізують можливість самоперевірки знань та визначення рівня їх засвоєння.

Загальний обсяг бакалаврської роботи 50 сторінок.

**ABSTRACT**

Qualifications Bachelor task is to develop mobile applications for testing the discipline "Organization of networks" that should help students in preparing for different types of tests, tests and more.

During the task the analysis of the subject area deals with methods for implementing control of knowledge and analogs are partially implement them. Also, analyzed using the tools that will be given the opportunity to correct implementation.

Mobile app testing requirements developed specification that describes the characteristics of the application in general, more detailed set of system characteristics, functional and nonfunctional requirements for the implementation of performance. Also, subsystem implemented Web service API, which should enable access to remote databases.

Based on system specifications, and considered unique domain generally performed system design. The basis for the implementation of the system architecture was selected pattern (pattern) design MVVM (Model View ViewModel).

To write a mobile app selected programming language C # 6.0 and SQLite library as a database for local database. To implement a Web service API uses Php Framework Symfony library and API Platform. To generate the client side to fill the remote database uses a markup language HTML and cascading style sheets CSS.

The task is successful and the mobile application from a web service can be used to assist students in preparing for exams, tests, etc., realize the possibility of self-knowledge and the definition of their assimilation.

The total volume of baccalaureate work 50 pages.

**ЗМІСТ**

[Вступ 8](#_Toc483348868)

[Розділ 1. Огляд існуючих аналогів та засобів розробки мобільних додатків 10](#_Toc483348869)

[1.1. Використання мобільних технологій у сьогоденні 10](#_Toc483348870)

[1.2. Огляд існуючих аналогів 12](#_Toc483348871)

[1.3. Засоби для розробки мобільних додатків 14](#_Toc483348872)

[1.4. Висновки 17](#_Toc483348873)

[Розділ 2. Постановка задачі розробки мобільного додатку для тестування знань студентів з дисципліни «Організація комп’ютерних мереж» 18](#_Toc483348874)

[2.1. Постановка задачі 18](#_Toc483348875)

[2.2. Специфікація вимог до мобільного додатку для тестування знань студентів 19](#_Toc483348876)

[2.3. Вибір інструментальних засобів для реалізації додатку 24](#_Toc483348877)

[Розділ 3. Проектування мобільного додатку тестування знань студентів «Організація комп’ютерних мереж» 27](#_Toc483348878)

[3.1. Вибір архітектури додатку 27](#_Toc483348879)

[3.2. Проектування графічного інтерфейсу користувача 31](#_Toc483348880)

[Розділ 4. реалізація та тестування мобільного додатку для тестування знань студентів з дисципліни «організація комп’ютерних мереж» 33](#_Toc483348881)

[4.1. Опис функціональності розробленої системи 33](#_Toc483348882)

[4.2. Програмна реалізація системи 34](#_Toc483348883)

[4.3. Тестування розробленої системи 41](#_Toc483348884)

[Розділ 5. економічна частина 44](#_Toc483348885)

[5.1. Економічна характеристика проектного рішення (програмного продукту) 44](#_Toc483348886)

[5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення проектного рішення (програмного продукту) 44](#_Toc483348887)

[5.3. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ 45](#_Toc483348888)

[5.4. Формування стратегічних альтернатив 47](#_Toc483348889)

[5.5. Бюджетування 47](#_Toc483348890)

[5.6. Вибір стратегії 52](#_Toc483348891)

[Висновки 54](#_Toc483348892)

[Список літератури 55](#_Toc483348893)

[Додаток А. Додаткові UML діаграми 58](#_Toc483348894)

[Додаток Б. Інструкція користувача мобільного додатку для тестування знань студентів з дисципліни «Організація комп’ютерних мереж» 60](#_Toc483348895)

[Додаток В. Звіт про тестування мобільного додатку для тестування знань студентів з дисципліни «Організація комп’ютерних мереж» 62](#_Toc483348896)

[Додаток Г. Програмний код мобільного додатку (реалізація класу QuestionNotify) 64](#_Toc483348897)

[Додаток Д. Налагоджувальний код контейнерів docker 68](#_Toc483348898)

[Додаток Е. Програмний код абстрактного класу AbstractApiController (реалізація функцій API) 69](#_Toc483348899)

Вступ

У даний час триває процес стрімкого наповнення освіти новітніми засобами інформаційних технологій, які вимагають наявність більш якісної оцінки знань, використовуючи усі сучасні методи навчального процесу. Стрімкий розвиток технологій, що не дає можливості призупинитись, миттєво наповнює життя людей новими можливостями, які набагато полегшують трудову діяльність та усі існуючі галузі господарювання.

У системі середньої спеціальної та вищої освіти все більш важливою стає комп’ютеризація та інформатизація. Сьогодні неможливо уявити наше життя без новітніх технологій. Технологій, які допомагають якісніше передавати, засвоювати та перевіряти матеріал. Тому питання впровадження та розробки електронної форми тестування знань є актуальним при підвищенні ефективності та якості навчання.

Не тільки у вищих навчальних закладах України, але і за кордоном стрімко відбувається розвиток систем освітнього процесу, та впроваджуються оцінювання досягнень студентів за допомогою електронних тестів. Тому все більше можна переконатись у актуальності даної теми [1].

Головну роль під час організації навчального процесу на основі будь-якої освітньої технології відіграє контроль отриманого рівня знань. Процедура контролю засвоєння знань виконує функцію констатації факту з метою перевірки якісного рівня готовності студента.

Під електронним контролем розуміють процес, що надає можливість оцінити рівень засвоєння матеріалу з метою керування поточним навчальним процесом та забезпечення індивідуального підходу до навчання кожного студента в електронному середовищі.

Необхідно відмітити, що саме для викладачів електронний контроль знань студентів має суттєвий перелік переваг над традиційною формою контролю якості знань:

* Дає можливість швидкої перевірки знань у великої кількості студентів.
* Звільняє викладача від виконання рутинної роботи та організації масового контролю, та додає більше часу для вдосконалення професійної діяльності.
* Надає можливість збільшити можливість реалізації індивідуальної роботи зі студентами, так як вагому частину знань студент змушений отримати самостійно.

Написання програмного забезпечення за темою «Мобільний додаток для тестування студентів “Організація комп’ютерних мереж”» зможе надати можливість реалізувати усі вище зазначені переваги та допомогти студентам полегшити процес вивчення, засвоєння та самоперевірки знань.

1. Огляд існуючих аналогів та засобів розробки мобільних додатків
   1. Використання мобільних технологій у сьогоденні

Розширення меж використання мобільних та веб технологій за останні 10 – 15 років зросло у декілька разів та стає стійкою тенденцією XXI століття. Для педагогів України це питання стало актуальним після того як використання цих технологій стало буденною справою кожного третього українця, у наслідок чого з’явилися різноманітні засоби масового тестування як у веб так і на мобільних платформах. Також з’явились механізми сертифікації, атестації та акредитації вузів у його нинішній формі, що містить елементи технологій тестування учнів та студентів.

На сьогоднішній день використання мобільних технологій присутнє на кожному кроці. У кожній з галузей використання різноманітної мобільної техніки не є дивним. Також кажуть XXI ст. є, століттям стрімкого розвитку інтернет технологій, та не можливо не замітити, що мобільні та веб технології тісно переплелись у сьогоденні.

У галузі освіти мобільні технології також найшли своє місце, та тісно переплелись з навчальним процесом.

Одним з найважливіших компонентів навчальної системи і одним з етапів навчального процесу є контроль за навчальною діяльністю, призначений для визначення успішності у навчанні кожного учня та студента. Аналізуючи отримані результати для корекції подальшого процесу навчання, підвищення успішності та якості засвоєння знань [2].

Тест це система яка базується на специфічній формі завдань з відповідним змістом та є обґрунтованим з наукової точки зору методом оцінювання знань, умінь та навичок учнів та студентів.

Допомагає у досягненні більш ефективного індивідуального та масового контролю навчально-виховним процесом. У порівнянні з традиційними формами контролю знань (іспит, залік, контрольна робота тощо) тестування неодноразово виявляється більш ефективним. Недостатньо всього лиш оцінити рівень знань, потрібно дізнатися як студент зможе використати ці знання. Тобто, зростає роль психологічного тестування.

Переваги тестової форми контролю знань над іншими були показані на практиці [3]:

* упродовж досить обмеженого часу може бути перевірена якість знань, навичок у зазначеної кількості студентів;
* можливий контроль знань, умінь, навичок на необхідному, заздалегідь запланованому рівні;
* реальним є самоконтроль;
* знання оцінюють більш-менш об’єктивно;
* увага студента фіксується не на формуванні відповіді, а не осмисленні її суті;
* створюють умови для постійного зворотного зв’язку між студентом і викладачем.

Проте тестовий контроль знань має й істотні недоліки:

1. ймовірність випадкового вибору правильної відповіді;
2. можливість при застосуванні тестів закритого типу оцінити тільки кінцевий результат (правильно - неправильно), у той час як сам процес, що привів до нього, не розкривається;
3. психологічний недолік – стандартизація мислення без врахування рівня розвитку особистості;
4. велика затрата часу на складання необхідного "банку" тестів, їх варіантів, трудомісткість процесу;
5. тести не сприяють розвитку мови.

Інформація в питанні тесту може подаватись у будь-якій формі – тексту, графічного зображення, звукового повідомлення, відео сюжету, формули тощо. Причому, введено в практику відкритість та доступність бази тестів, тобто зміст тестів відомий студентам. Але при цьому тести мають постійно поновлюються з урахуванням змін у навчальних планах, програмах, реаліях сьогодення.

Дуже важливим при проведені тестового контролю є дотримання організаційного моменту (пояснення мети, порядку виконання та оформлення тесту, визначення часу та його виконання, забезпечення кожного студента бланком відповідей стандартного зразка, що великою мірою заощаджує час і студента, і викладача).

Тестові завдання значно скорочують час очікування студентами оцінки, що є суттєвим психологічним та виховним фактором.

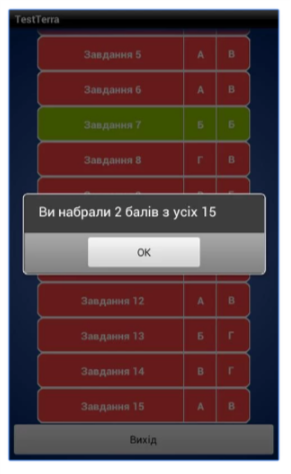
Після тестування, а воно може охоплювати матеріал окремої теми, модулю або курсу в цілому, обов’язково аналізують його результати. Аналіз необхідний для того, щоб студент зміг перевірити, наскільки адекватно він оцінює свої знання, повірити у власні сили і скорегувати свою підготовку [4].

Важливою умовою тестування, як універсального педагогічного інструмента, є частота його проведення, яка залежить від дисципліни, її ролі і місця в навчальному плані, особливостей засвоєння знань. Слід зробити тестування звичною і зручною формою регулярного контролю знань студентів. Необхідно пам’ятати, що тестування – це не самоціль, а ефективна форма повторення – узагальнення і впорядкування вивченого. Контрольно-оцінювальна функція навчання – це лише елемент добре організованого і технологічно продуманого навчально-виховного процесу. Якщо студенти матимуть міцні знання, то їх оцінювання не становитиме особливих труднощів, в якій би формі воно не проводилось [5].

* 1. Огляд існуючих аналогів

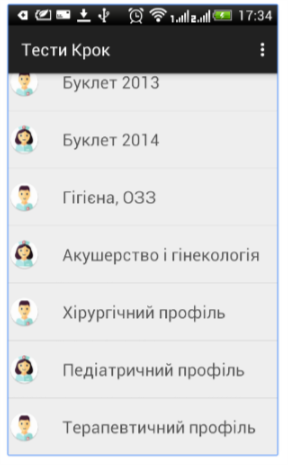
Для особистого тестування та вивчення різноманітного матеріалу вже створено велика кількість різноманітних веб та мобільних додатків. Прикладами яких є:

1. «TestTerra (ЗНО)» – Програма для підготовки до ЗНО (Зовнішнє незалежне оцінювання). Розроблена для абітурієнтів, які готуються до вступу у вищий навчальний заклад(ВНЗ). Програма містить тести для підготовки до ЗНО з української мови та української літератури, Історії України. Тести сертифіковані та надані ЛРЦОЯО (Львівський Регіональний Центр Оцінювання Якості Освіти) [6]. Позитивними характеристиками даного додатку є можливість самоперевірки знань та збереження результатів тестувань. Недоліками ж є відсутність можливості перевірки оновлення бази даних запитань (рис. 2.1).



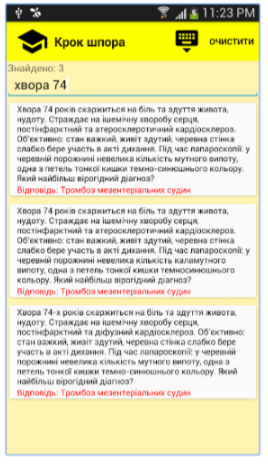
* + 1. Додаток «TestTerra (ЗНО)»

1. «Тести Крок» – Тести Крок - додаток Android, що дозволяє підготуватись до здачі обов’язкового державного тестування Крок студентам медичних університетів України. Дана версія містить буклети і базу тестів загального профілю для здачі Крок 2 та Крок 3 у 2016 році [7]. Позитивними характеристиками додатку можна відзначити такі як: повна збірка питань з останніми оновленнями, приємний інтерфейс користувача. Негативними характеристиками є такі: відсутність збереження результатів (рис. 2.2).



* + 1. Додаток «Тести крок»

1. «Крок Помічник» – Програма для допомоги у швидкому вивченні питань до екзамену «Крок» у вищих медичних закладах. Надає можливість пошуку потрібного питання та відповіді до нього [8]. Позитивними характеристиками додатку є приємний інтерфейс користувача та можливість пошуку потрібного питання. Недоліками даного додатку є відсутність можливості проходження тестування та оновлення бази даних питань (рис. 2.2).



* + 1. Додаток «Крок помічник»

У наведених аналогах відображаються основні недоліки додатків даного напряму, тобто результати тестувань або підготовку студента може переглядати тільки студент не надаючи можливості викладачу переглянути результати.

* 1. Засоби для розробки мобільних додатків

На даному етапі розвитку сучасності існує велика кількість різноманітних технологій, які надають можливість створювати мобільні застосунки.

Основним кроком на початку розробки мобільного додатку є вибір операційної системи. Основними операційними системами сучасності є:

1. Android - гнучка операційна система для смартфонів і планшетів. Техніка на базі Android зараз перебуває на етапі стрімкого оновлення, тому розробка мобільних додатків, розміщених на Google Play, користується особливою популярністю.
2. iOS (відома як iPhone OS) — це мобільна операційна система від Apple для смартфонів, планшетів, плеєрів та телевізійних приставок. Надає можливість завантаження різноманітних мобільних додатків з спеціалізованого інтернет магазину App Store, який станом на 31 травня 2011 року містить понад 500 тисяч застосунків.
3. Windows Phone – операційна система яка є масштабним оновленням Windows Mobile. Серед безліч одноманітних телефонів і мобільних пристроїв дана операційна система виділяється своєю динамікою. Також надає можливість завантаження додатків з інтернет магазину.

Наступними кроками у розробці є створення прототипу та проектування структури UI / UX дизайну. На цьому етапі розглядається весь функціонал, виконуються усі функції, що прописані у технічному завданні, визначається як буде працювати додаток та які елементи керування будуть розміщені на кожному з екранів.

Для створення прототипів існує велика кількість онлайн сервісів та програмного забезпечення. Вибір сервісу або пз залежить від потреб.

Далі відштовхуючись від прототипів екранів, функціоналу та цільової аудиторії створюється дизайн, опрацьовується кожен екран та найдрібніші його деталі.

Після моделювання зовнішнього вигляду додатку, у роботу залучаються розробники та архітектори функціоналу. Відбувається розробка взаємодії дизайну та логічної частини.

Середовище та мова програмування залежить від вибраної операційної системи. Прикладами середовища є Eclipse, XCode, Visual Studio і т.д. Кожна з операційних систем та середовище програмування підтримує роботу з визначеними мовами програмування. Основними мовами програмування для мобільних додатків є:

1. HTML5 – мова програмування для розробки веб-орієнтованих додатків. Підтримка стандарту HTML5 забезпечується практично на всіх мобільних платформах, включаючи iOS, Android и Windows Phone, що являється важливою перевагою.
2. Java – мова програмування в основному направлена на розробку для мобільної платформу Android, але це не обмежує її можливості. Однією з особливостей мови Java є її універсальність. Маючи можливість компілюватись, мова може працювати в двох режимах: в вікні браузера, або запускатись на віртуальній машині.
3. C++ – об’єктно-орієнтована мова, яка може бути використана для розробки майже на всіх платформах і для будь яких цілей. Забезпечує модульність, роздільну компіляцію, обробку виключень, абстракції даних, віртуальні функції. Поєднує властивості високо та низько рівневих мов програмування.
4. Objective C – мова програмування похідна від мови C як і С++, але орієнтована для розробки програмного забезпечення пристроїв компанії Apple. Являється частиною середовища розробки Apple та повністю інтегрована у всі фреймворки IOS та MacOS.
5. Swift – компільована мова програмування розроблена компанією Apple. Підтримує багатопарадигмову розробку. Компілятор Swift побудований з використанням технологій вільного проекту LLVM. Успадкувала найкращі елементи мов C та Objective-C , тому синтаксис є дуже схожий на синтаксис усіх похідних мов C, але водночас відрізняється використанням засобів автоматичного розподілу пам’яті і контролю переповнення змінних і масивів, що збільшує безпеку та надійність коду.
6. C# – мова програмування розроблена компанією Microsoft, також є схожою на мови C та Java, є об’єктно-орієнтованою з безпечною системою типізації для платформи .NET.

Кожен крок даного етапу підлягає тестуванню та детальній перевірці. На при кінці розробки функціоналу також проводиться тестування але усієї систему в цілому.

Звісно для того щоб клієнти могли скористатись додатком, його розміщують у спеціальних магазинах, наприклад: GooglePay, AppStore або Windows Mobile Store. Вибір магазину залежить від цільової аудиторії та техніки якою вона користується.

* 1. Висновки

Для виявлення потреб користувачів були розглянуті аналоги подібних мобільних додатків які описані у пункті 1.2 огляд існуючих аналогів. Кожен з аналогів містить як позитивні так і негативні характеристики основними з який є:

* Відсутність можливості збереження результатів.
* Відсутність перевірки оновлення бази даних питань.
* Відсутність надати викладачу можливість переглянути результати.
* Відсутність більш детального вибору курсу, модулю або теми.

Також виходячи з аналізу, було вирішено обрати платформу .Net Windows Mobile, на якій аналогів знайдено не було.

Тому було вирішено розробити систему яка буде складатись з двох підсистем (веб сервіс та мобільний додаток) та максимально буде вирішувати недоліки аналогів. Метою системи є допомогти студенту у підготовці та самоперевірці знань, з можливістю збереження результатів, перегляду статистики та надання викладачу можливість переглянути результат. Також як джерело питань для мобільного додатку буде використовуватись віддалена база даних. Для наповнення бази даних буде реалізовано веб сервіс за допомогою якого викладач зможе наповнювати базу даних.

1. Постановка задачі розробки мобільного додатку для тестування знань студентів з дисципліни «Організація комп’ютерних мереж»
   1. Постановка задачі

Завдання бакалаврської роботи є проектування, розробка та тестування мобільного додатку для перевірки знань з дисципліни «Організація комп’ютерних мереж». З його допомогою користувач (студент) зможе самостійно провести самоперевірку своїх знань по обраному курсу, модулю або темі та успішно підготуватись до іспитів, контрольних або інших тестових робіт.

Для викладача додатково буде розроблено API сервіс для доступу до віддаленої бази даних, у якій повинні зберігатись результати тестувань студентів, які з мобільного додатку будуть вивантажуватись. Також викладач за допомогою API сервісу має змогу редагувати данні у віддаленій базі даних.

Тому для рішення поставленої задачі потрібно реалізувати дві підсистеми: мобільний додаток та API сервіс.

Мобільний додаток має мати таку функціональність:

1. Авторизація користувачів

Відправка запиту до API для перевірки чи є такий користувач та його авторизації у системі.

1. Можливість завантаження віддаленої бази даних на телефон для роботи з питаннями.

Під час першого запуску додатку буде відправлятись запит до API для отримання записів з таких таблиць (курс, модуль, тема, питання, відповіді, еквівалентні питання), що дасть змогу сформувати локальну базу даних. Після чого користувач зможе проходити тестування. Надалі користувач зможе перевіряти віддалену базу даних на оновлення та при бажанні завантажувати їх.

1. Можливість проходження тестування по вибраному: курсу, модулю або темі.

Вибравши необхідний курс, модуль або тему відбувається вибірка питань та перехід не екран де будуть послідовно відображатись питання та варіанти відповідей або поле для введення відповіді. Перевірка правильності відповіді буде відбуватись одразу після переходу на наступне питання. Після проходження тестування користувачу буде відображатись результат та записуватись у локальну базу даних.

1. Перегляд статистики.

Вибірка тестів та відображення результатів по курсам, модулям та темам.

Реалізація веб сервісу API необхідна для постійного оновлення віддаленої бази даних, а саме оновлення питань та інших даних та перевірка на відповідність до навчального плану.

Веб сервіс API повинен мати наступний функціонал:

1. Реалізація доступу до бази даних.

Створення контролерів, моделей даних та веб представлення для надання доступу до бази даних.

1. Авторизація викладачів.

Перевірка наявності користувача з веб частини, та ведених логіну та паролю на вірність.

1. Можливість перегляду результатів тестування студентів.

Відправка запиту до бази даних для перегляду результатів студентів.

* 1. Специфікація вимог до мобільного додатку для тестування знань студентів

1. **Вступ**
   1. *Призначення, мета*

Мобільний застосунок для тестування знань студентів. Даний продукт є системою, яка складається з мобільного додатку та веб сервісу (API) для роботи з віддаленою базою даних.

* 1. *Продукти-аналоги*

«TestTerra (ЗНО)» – Програма для підготовки до ЗНО (Зовнішнє незалежне оцінювання) [4]. Відмінностями даного застосунку є відсутність веб частини, та потреби у підключенні до мережі інтернет. Також можна зазначити відсутність оновлення бази даних та відображення статистики.

«Тести Крок» – Тести Крок - додаток Android, що дозволяє підготуватись до здачі обов’язкового державного тестування Крок студентам медичних університетів України [6]. Даний застосунок має таку відмінність як відсутність веб частини та не розділяється на підсистеми, та не потребує підключення до мережі інтернет. Також відсутня можливість збереження та перегляду статистики тестування.

«Крок Помічник» – Програма для допомоги у швидкому вивченні питань до екзамену «Крок» у вищих медичних закладах [5]. Під час аналізу даного додатку виявлено відмінність у відсутності можливості самоперевірки знань, статистики пошуку питань, відсутність підключення до мережі інтернет та розділення на наявність веб підсистеми.

1. **Загальний опис**
   1. *Характеристики продукту*

Даний застосунок може надавати студентам можливість готуватися та проходити самоперевірку знань. Вибравши потрібні курси, модулі або теми студент може розпочати тестування. В кінці тестування студенту буде відображено результат, тобто кількість всіх питань та кількість питань на які студент відповів правильно. Також результати будуть зберігатися, для подальшого перегляду та аналізу. При бажанні студент може перевіряти наявність оновлень та завантажити їх.

* 1. *Класи користувачів та їх характеристики*

Користувачі які будуть використовувати дану систему є студенти та викладачі.

Викладач буде мати змогу вносити, редагувати та видаляти записи. Також викладач може переглядати результати тестування. Доступ до системи викладачу буде надано у вигляді веб ресурсу.

Студент буде мати змогу авторизуватися в системі, проходити тестування по вибраним курсам, модулям або темам та переглядати результати власного тестування на мобільному пристрої.

* 1. *Середовище функціонування*

Мобільна частина системи буде функціонувати на платформі Windows Mobile, під керівництвом операційної системи Windows Phone 8.1. Локальна база даних буде реалізована за допомогою розширення SQLite.

Веб частина буде реалізована за допомогою PHP Framework, який буде розміщений на сервері з операційною системою Linux.

Віддалена база даних буде реалізована за допомогою СУБД MySQL, яка також буде розміщена на сервері з операційною системою Linux.

1. **Характеристики системи**
   1. *Реєстрація користувача*
      1. *Опис і пріоритет*

Користувач може реєструватися у системі, в результаті йому надається можливість проходити тестування. Пріоритет – високий.

* + 1. *Послідовності дія/відгук*

Запустити додаток, якщо додаток запущений вперше натиснути кнопку реєстрація, далі ввести інформацію про користувача у відображені поля та натиснути зареєструватись.

* + 1. *Функціональні вимоги*
       1. У серверній частині потрібно перевірити введені данні на наявність такого користувача.
       2. Данні про користувача зберігаються у віддаленій та локальній базі даних.
       3. Користувачу мають відображатися поля для введення.
       4. В базі даних потрібно передбачити шифрування паролю.
  1. *Проходження тесту*
     1. *Опис і пріоритет*

Користувач може пройти тестування, в кінці тестування відображається результат. Пріоритет – високий.

* + 1. *Послідовність дія/відгук*

Запустити додаток, обрати потрібний курс натискаючи на ліву межу елементу списку або вибравши пункт меню «вибрати» для встановлення міток та натиснути на кнопку «тестування» для відображення питань, в кінці тестування користувачу буде відображено результат.

* + 1. *Функціональні вимоги*
       1. Результати тестувань зберігаються у локальну та віддалену базу даних.
       2. Користувачу має відображатись питання та варіанти відповідей з можливістю вибору або введення відповіді.
       3. У кінці кожного тесту має відображатись результат.
  1. *Перегляд результатів тестування*
     1. *Опис і пріоритет*

Відображає весь список результатів з можливість сортування по типу та даті проходження тестування. Пріоритет – середній.

* + 1. *Послідовність дія/відгук*

Запустити додаток, вибрати пункт меню «результати тестування», для сортування списку вибрати пункт меню «сортувати по типу» або «сортувати по даті».

* + 1. *Функціональні вимоги*
       1. Користувачу має відобразитись список результатів з локальної бази даних
       2. Для сортування мають бути реалізовані пункти меню

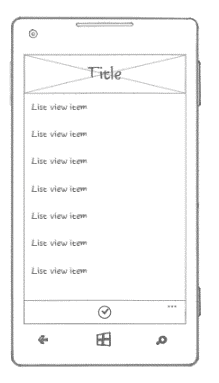
1. **Вимоги зовнішніх інтерфейсів**
   1. *Користувацькі інтерфейси*

Користувацький інтерфейс буде розподіллено на дві частини перша з яких буде інтерфейсом мобільного додатку та друга інтерфейс веб додатку.

Інтерфейс мобільного додатку буде складатись з набору екранів.

1. Список курсів (відображає список курсів з можливістю вибору курсів для проходження тестування).
2. Список модулів (відображає список модулів з можливістю вибору та проходження тестування).
3. Список тем (відображає список модулів з можливістю вибору та проходження тестування).
4. Екран тестування (відображає питання з вибраних курсів або модулей або тем у випадковій послідовності та варіанти відповідей до кожного відповідно).
5. Екран статистики (відображає список пройдених тестів та їхні результати).
6. Екран авторизації (відображає поля для введення логіну та паролю для проведення перевірки на наявність користувача в базі даних та відповідність логіну та паролю).
7. Екран реєстрації (відображає поля для введення інформації нового користувача).

Прикладом користувацького інтерфейсу є прототип екрану списку (списку курсів, модулей та тем які на рівні прототипу однакові) (рис. 3.1).



* + 1. Прототип екрану списку (курсів, модулей та тем)
  1. *Апаратні інтерфейси*

Апаратними інтерфейсами у даній системі є мобільний телефон з операційною системою Windows Phone 8.1 та сервери з API та базою даних MySQL з операційною системою Linux. Зв’язки між інтерфейсами будуть наступні:

1. Мобільний додаток звертатиметься до API за комунікаційним протоколом HTTP
2. API звертатиметься до бази даних MySQL по протоколу TCP
   1. *Програмні інтерфейси*

Даний продукт буде працювати з різними програмними компонентами, а саме:

1. Локальна база даних (використовує бібліотеку «SQLite.Net-PCL» версії 3.1.1 для роботи з файлом бази даних)
2. Для деяких елементів керування буде використовуватись розширення «WPtoolkit» версії 4.2013.08.16
3. Для відділеної бази даних використовуватиметься «MySQL Server» версії 5.6
4. **Інші не функційні вимоги**
   1. *Вимоги продуктивності*

Вимогою для забезпечення продуктивності системи є надання доступу до інтернету, для можливості реєстрації, авторизації та завантаження бази даних на телефон.

* 1. *Вимоги безпеки*

Для захисту інформації реалізовуватиметься система авторизації для користувачів, з підвищеною надійністю та шифруванням паролей.

Для безпечного доступу з мобільного додатку буде реалізовано використання окрім логіну та паролю API ключ який буде таємним для кожного користувача.

* 1. Вибір інструментальних засобів для реалізації додатку

Даний проект складається з двох підсистем (мобільний додаток та веб сервіс), які будуть функціонувати в різному апаратному середовищі, під управлінням різних операційних систем тому інструментальні засоби для кожної підсистеми будуть кардинально відрізнятись.

Для реалізації функціональності мобільного додатку за основу вибрано Windows Phone SDK, що є частиною платформи .NET Framework та повнофункціональним середовищем для розробки програмних засобі та ігор для Windows Phone 8.0 – 8.1 [9].

Дане середовище має наступні вимоги до системи:

* ОС Windows 8, 8.1, 10
* 6.5 Гб вільного дискового простору
* 64-розрядний ЦП (x64)
* Підтримка віртуалізації ЦП (для підтримки роботи емулятора, якщо підтримка віртуалізації відсутня тоді пакет Windows Phone SDK буде встановлено але запуск емулятора буде не можливий тому потрібно підключати додаткові пристрої)

Одним з основних компонент для роботи мобільного додатку є локальна база даних. Для роботи з базою даних використовується полегшена реляційна система керування базами даних SQLite. Реалізація її можливостей на у даній підсистемі забезпечується пакетом SQLite-net PCL версії 1.3.1. В основу SQLite-net PCL лягає бібліотека sqlite3.dll яка на низькому рівні має повний доступ до файлу бази даних та роботи з ним.

Також для забезпечення стабільності та наявності додаткового функціоналу до основного пакету по роботі з базою даних, підключаються наступні:

* SQLite.Net.Async-PCL – забезпечує асинхронність роботи бази даних, надає додаткові атрибути для оголошення таблиць.
* SQLite.Net.Core-PCL – являється ядром основного пакету, забезпечує роботу на різних платформах (Windows 8, Windows Phone 8, Android, Win32 тощо).
* sqlite-net-wp8 – являється розширенням пакету та деяким посередником в реалізації роботи з типом Blob. Також є частковим розширенням по роботі з файлом, Реалізоване на мові C++/CX.
* SQLitePCLRaw.lib.e\_sqlite3.v110\_wp80 – містить специфікацію пакету по роботі з базою даних та інструменти по роботі з нею [10].
* SQLitePCLRaw.provider.e\_sqlite3.wp80 – SQLitePCL.raw 'provider' зв’язує SQLitePCLRaw.core і конкретний екземпляр бібліотеки SQLite. В залежності від платформи додатково встановлює один з пакетів SQLitePCLRaw.lib.\* [11].

Саме цей набір пакетів та бібліотек забезпечує надійний зв'язок з локальною базою даних. Містить всі необхідні інструменти та є надійним ліцензованим продуктом, що має детальну документацію та широко використовується на платформі .Net Framework.

Для підвищення зручності та інтуїтивності користуватського інтерфейсу, було підключено пакет WPtoolkit версії 4.2013.08.16. Даний пакет представляє собою набір розширених елементів керування, які надають користувачу додаткові можливості та зручність у використанні [12].

У проекті був використаний один з елементів керування даного пакету LongListMultiSelector, що представляє з себе дочірній елемент керування ListBox (відображення списку). Він розширює можливості керування вибору та відображення елементів.

Для реалізації функціональності веб сервісу (серверної частини), буде використано Docker як середовище яке буде складатись з контейнерів [13]. Тобто Docker представляє набір віртуальних машин які взаємодіють між собою. Контейнери являються відносно не залежними та водночас надають одне одному певні можливості та ресурси. Будуть реалізовані такі контейнери як:

* Testing\_app – об’єднує в собі контейнери Php та Nginx.
* Testing\_nginx – являється сервером для Php, містить в собі налаштування та інші відомості та можливості серверу.
* Testing\_php – є основним для збереження коду та містить в собі весь набір вбудованих в Php та додаткових бібліотек.
* Testing\_mysql – є сервером для бази даних, та містить в собі саму базу даних.

Для взаємодії контейнерів на рівні протоколу TCP кожен з них має IP адресу та порт. Також для додаткових можливостей можна відкрити доступ до будь якого іншого порту в кожного з контейнерів. Для реалізації функціоналу описаному у пункті 3.1 буде використано Php Micro Framework, який буде розміщений у контейнері Testing\_php. У Php Framework буде відбуватись доступ до бази даних та робота з нею. Також буде реалізовано набір методів для завантаження записів з бази даних у базу даних мобільного пристрою.

1. Проектування мобільного додатку тестування знань студентів «Організація комп’ютерних мереж»

Для графічного опису загальної структури системи буде використано UML діаграми. За допомогою діаграм дається можливість розглянути поведінку об’єктів, їх структуру, взаємодію між собою та середовищем [14].

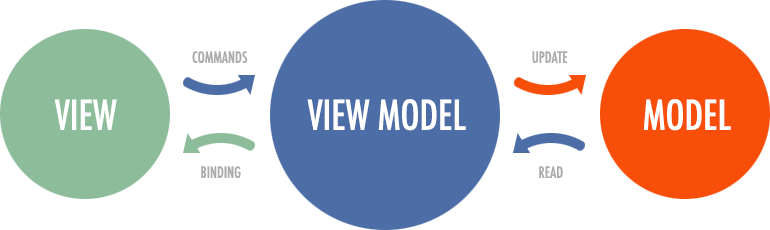
* 1. Вибір архітектури додатку

Для реалізації додатку буде використано шаблон проектування MVVM (Model View ViewModel) [15].

MVVM зручно використовувати замість класичного MVC та йому подібних у тих випадках, коли на платформі де ведеться розробка, присутнє «зв'язування даних».

В MVC/MVP зміни у користувацькому інтерфейсі не впливають безпосередньо на модель, а йдуть через Контролер. У таких технологіях, як WPF та Silverlight, присутня концепція «зв'язування даних», що дозволяє зв'язувати дані із візуальними елементами в обидві сторони [16].

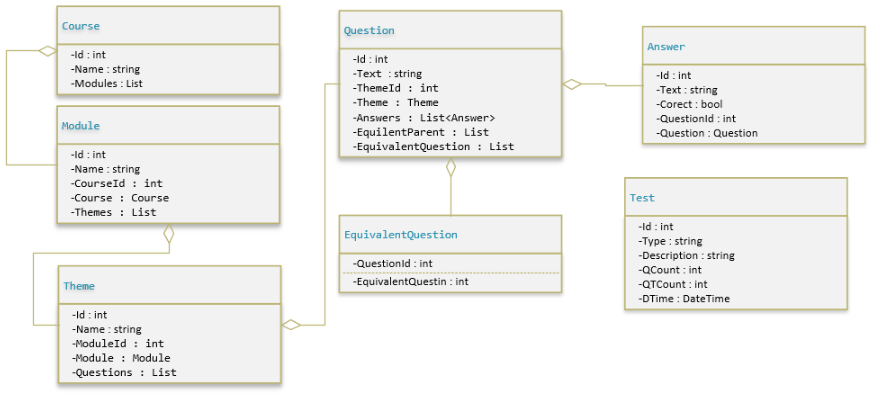
Використання даного шаблону проектування дає можливість розділити модель даних та користувацький інтерфейс через об’єкт посередника (ViewModel) (рис. 3.1).



* + 1. Схематичне зображення шаблону  
       MVVM (Model View ViewModel)

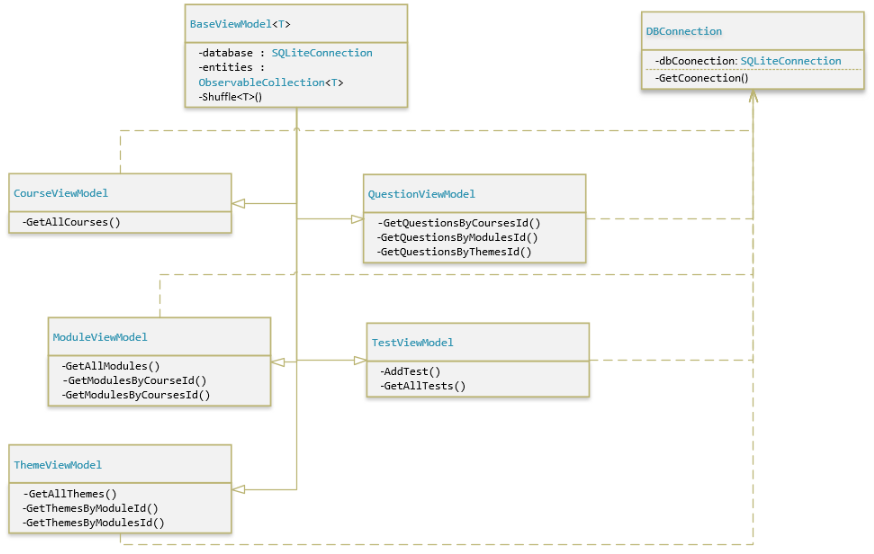
Також буде використано шаблон проектування одинак (Singleton) у класі DBConnection. Даний шаблон гарантує, що клас матиме тільки один екземпляр, і забезпечує глобальну точку доступу до цього екземпляра. Це надає можливість мати єдине підключення до локальної бази, та усунення пов’язаних з цим колізій.

Базуючись на структурі шаблону MVVM класи поділяються на такі групи: моделі, представлення та модель представлення (рис. 3.2, рис. 3.3).



* + 1. Діаграма класів групи модель

Це надає допомогу розробнику у великих проектах та надає більш зрозумілий вигляд.

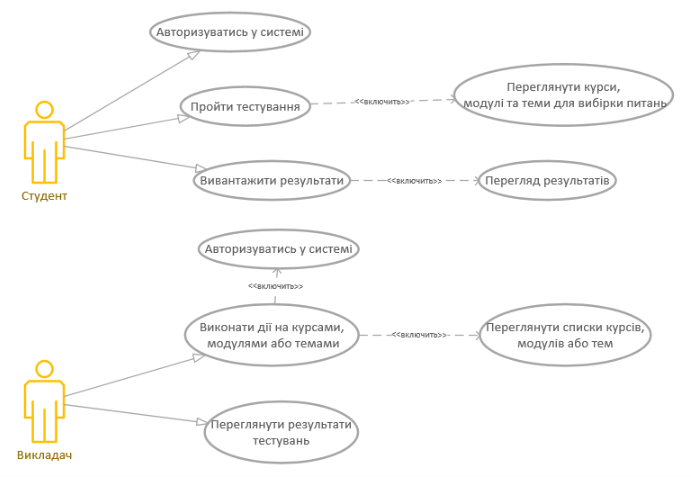


* + 1. Діаграма класів групи модель представлення

Такий метод розділення класів на групи допомагає у розробці та розумінні логіки системи. Також у даній системі допомагає при роботі з локальною базою даних, адже група класів модель є проекцією таблиць бази даних. Тобто кожен клас містить у собі усі поля таблиці та за допомогою допоміжних класів які визначають мета дані налаштовую властивості кожного поля (типи полів, назви полів, ключі, зв’язки з іншими таблицями).

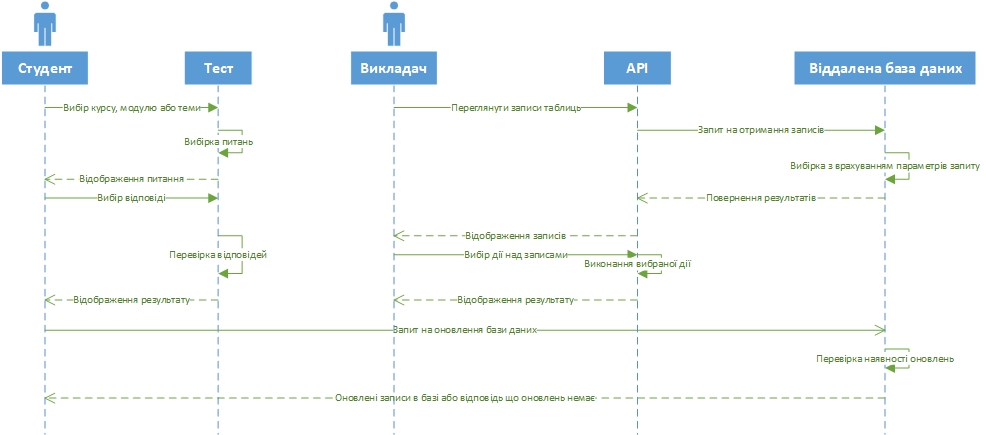
Для більш узагальненого представлення функціональності системи реалізується діаграма прецедентів (варіантів використання).

Дана система розділяється на підсистеми (Windows Phone Application, Server API), у кожної з яких є по одному актору. Таким чином можна функціональність можна розподілити між підсистемами які явно не з’єднані(рис. 3.4).



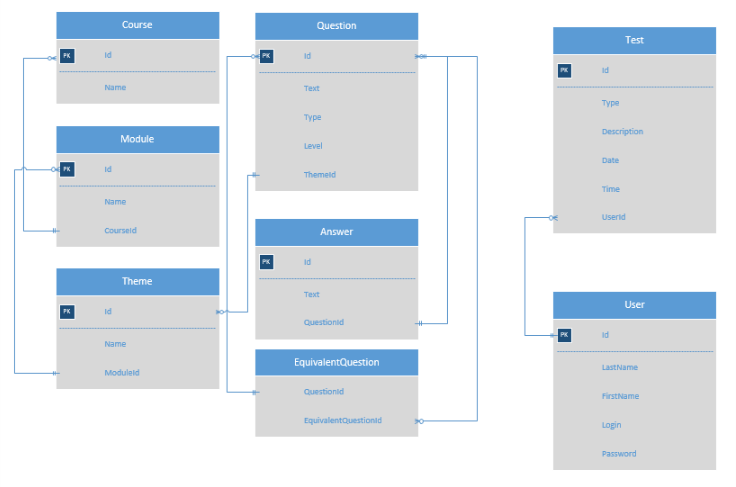
* + 1. Діаграма прецедентів (варіантів використання)

Для відображення взаємодії об’єктів на часовій шкалі використовується діаграма послідовності. У даному випадку на часовій шкалі буде зображено взаємодію таких акторів як студент та викладач з об’єктами тест, API та віддаленою базою даних (рис. 3.5).



* + 1. Діаграма послідовності

У кожній з підсистем проекту (мобільний додаток та веб сервіс) використовується база даних, для реалізації віддаленого тестування без доступу до мережі інтернет. В загальному моделі обох баз даних схожі не враховуючи потреби середовищ кожної (рис. 3.6).



* + 1. Загальна модель бази даних

На відміну від середовища MySQL, SQLite автоматично створює додаткову таблицю для збереження відомостей про базу даних та її структуру. За потребою зображення додаткових UML діаграм міститься у додатку А.

* 1. Проектування графічного інтерфейсу користувача

Метою розробки графічного інтерфейсу є надання користувачу зручних умов у роботі з додатком, інтуїтивне розміщення елементів управління та зрозуміле подання інформації.

У даному проекті інтерфейс користувача розробляється для мобільного пристрою на базі OS Windows Phone 8.1, тому проектування прототипів підлаштовується під можливості декларативної мови розмітки графічного інтерфейсу XAML [17].

Для реалізації вище зазначеного функціоналу підсистем (мобільного додатку та веб сервісу), набір екранів розділяється відповідно.

Екрани для мобільного додатку:

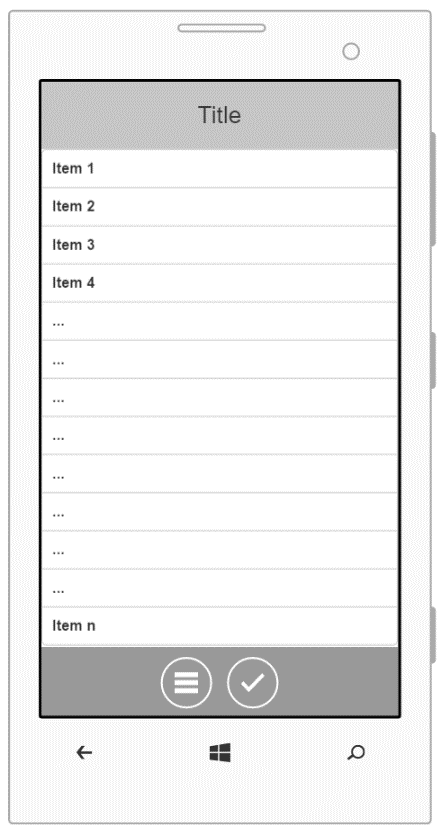
* Список курсів
* Список модулів
* Список тем
* Екран тестування
* Екран статистики
* Екран авторизації
* Екран реєстрації

Для екранів списків курсів, модулей та тем є однакові, тому що розміщення елементів та їх вигляд є однакові (рис. 3.7).

Екрани для веб сервісу:

* Екран авторизації
* Екран реєстрації
* Екран перегляду тем, модулів та курсів
* Екран перегляду результатів

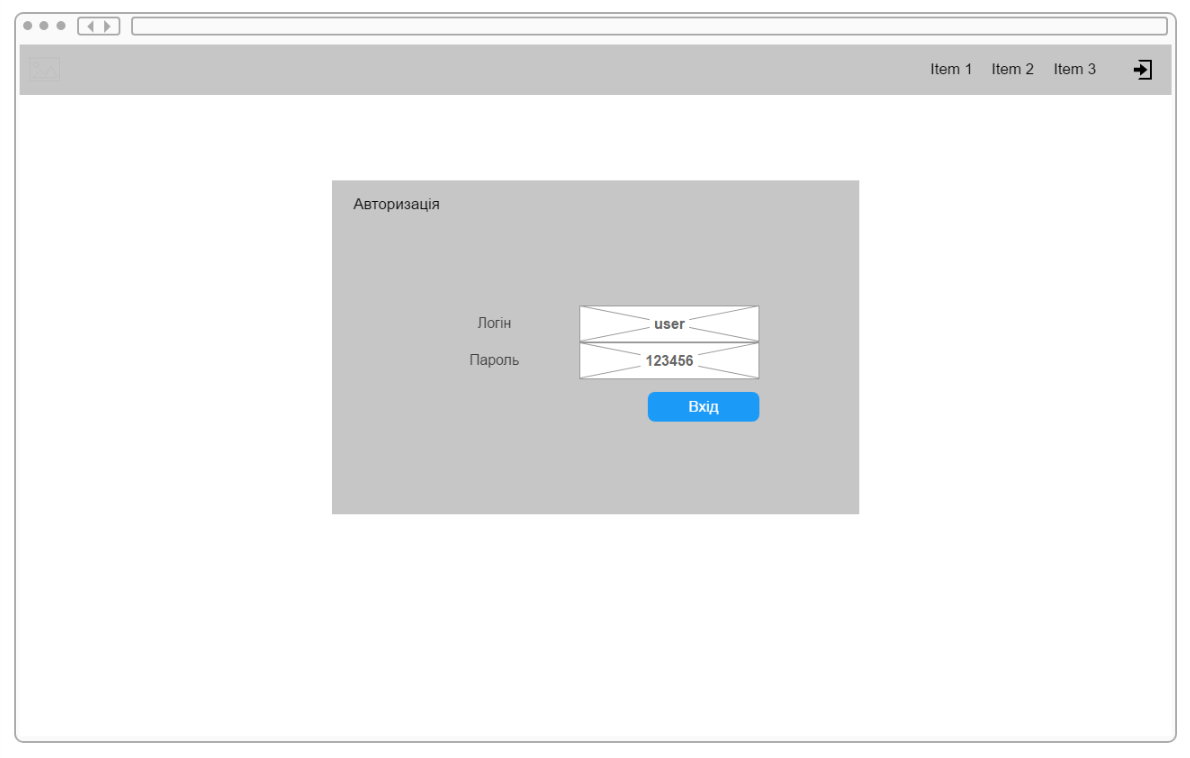
Першим з етапів проектування графічного інтерфейсу є прототипування, тобто створення прототипів (ескізів), для подальшої розробки графічного інтерфейсу засобами середовища програмування, у даному випадку мовою розмітки XAML [18].



* + 1. Прототип екранів списків

Оскільки друга підсистема проекту є веб сервіс та розміщення його буде на сервері, реалізація екранів буде здійснюватися за допомогою мови розмітки гіперпосилань HTML та каскадних таблиць стилів CSS.

Прикладом прототипу для екрану авторизації у веб сервісі є рис 3.8.



* + 1. Прототип екрану авторизації у веб сервісі

На рис. 3.8. зображено екран авторизації з полями для введення логіну та паролю, кнопка після натиску на яку відбувається перевірка введених даних та авторизація користувача.

1. реалізація та тестування мобільного додатку для тестування знань студентів з дисципліни «організація комп’ютерних мереж»
   1. Опис функціональності розробленої системи

Проект складається з двох підсистем (мобільний додаток та веб сервіс). Дана реалізація забезпечує розподілення навантаження та функціоналу, надає можливість реалізації множинного доступу до віддаленої бази даних.

* + - 1. Опис функціональності мобільного додатку

У функціональність мобільного додатку входить наступне:

* Авторизація користувача (студента) – під час кожного запуску додатку відбувається перевірка на наявність користувачів у таблиці з поміткою активний, якщо такий користувач знайдено усі результати тестувань будуть прив’язуватись до цього користувача, в іншому випадку відображається екран авторизації на якому можна вести логін та пароль або на цьому екрані натиснути кнопку пропустити. В такому випадку результати не будуть прив’язуватись та їх не можна буде вивантажувати у віддалену базу даних.
* Завантаження віддаленої бази даних – під час першого запуску додатку буде виконана перевірка на наявність підключення до мережі інтернет, та запуску набору запитів для заповнення таблиць.
* Проходження тестування за вибраним курсом, модулем або темою, також надається можливість вибрати одразу декілька курсів, модулей або тем. Після першого завантаження бази даних користувачу буде показано головний екран на якому буде відображатись список курсів. На ньому
* Збереження результатів тестувань – після кожного пройденого тестування результати зберігаються у локальній базі даних, для подальшого перегляду та вивантажування.
* Відображення результатів з можливістю сортування по типу та даті проходження тесту. Відкривши меню головного екрану та вибравши пункт «результати тестувань» можна перейти на екран результатів на якому будуть відображатись усі результати та інформація про них (назви курсів модулей або тем по яким виконане тестування, кількість питань у тесті, кількість правильних питань, дата та час проходження тесту та відмітка чи прив’язаний він до користувача). Також надається можливість вибрати необхідні результати та вивантажити їх у віддалену базу даних (з перевіркою на прив’язку до користувача).
  + - 1. Опис функціональності серверної частини

Функціональність серверної частини логічно поділяється на невеликі модулі дії яких є пов’язані, а саме:

* Rest API – до функціональності даного модуля входить можливість надавання віддаленого доступу до бази даних. Тобто виконання таких дій, як читання, запис, редагування та видалення записів у базі даних.
* Клієнтська частина для користувачів типу викладач – у функціональність входить надання можливості за допомогою браузера отримувати доступ до серверної частини:
  + Авторизація у системі – перевіряє користувача на наявність, вірність логіну та паролю та авторизує у системі з наданням прав на використання функцій API.
  + Перегляд записів – надає можливість використання функцій API для виконання дій над списками курсів, модулей, тем, питань, відповідей.
  + Додавання студентів – дає можливість додати студента у систему, по замовчуванню встановлює зв'язок з авторизованим користувачем.
  + Перегляд результатів – надає можливість переглянути результати тестувань студентів зв’язаних з авторизованим користувачем.
  1. Програмна реалізація системи
     + 1. Програмна реалізація мобільного додатку

Для реалізації мобільного додатку вибрано цільову платформу .Net, мову програмування C#. Мобільний додаток розроблений для пристроїв на базі операційної системи Windows Phone 8.1 [19, 20].

Для побудови архітектури вибрано саме шаблон проектування MVVM (Model View ViewModel), тому що він використовується у системах де відбувається прив’язка даних між екраном та класом роботи з моделлю даних. На основі архітектури, файлова структура додатку поділяється такі основні групи, як вигляд, модель даних, модель представлення що також чітко відображає логічну структуру. У даному випадку кожен з виглядів прив’язаний до відповідної моделі представлення.

У кожній з функцій мобільного додатку використовується підключення до локальної бази даних та робота з нею. Щоб отримати підключення до локальної бази даних (об’єкт класу SQLiteConnection), написаний наступний програмний код:

static class DBConnection

{

public static string DB\_NAME = "test.db";

public static string DB\_PATH = Path.Combine(ApplicationData.Current.LocalFolder.Path, DB\_NAME);

static SQLiteConnection dbCoonection;

public static SQLiteConnection GetCoonection()

{ if (dbCoonection != null)

{ return dbCoonection; }

else

{ SQLite.Net.Platform.WindowsPhone8.SQLitePlatformWP8 e = new SQLite.Net.Platform.WindowsPhone8.SQLitePlatformWP8();

dbCoonection = new SQLiteConnection(e, DB\_PATH);

return dbCoonection; }

}

}

У вище відображеному програмному коді описано, статичний клас у функції якого входить назви файлу бази даних, шлях до файлу бази даних та метод що перевіряє наявність підключення та у разі його відсутності створює нове в протилежному випадку повертає існуюче.

Для роботи з локальною базою даних необхідно реалізація кожної таблиці як класу, тобто створення проекції моделі даних. У реалізації кожного з класів моделі даних необхідно використати додаткові класи мета даних які налаштовують та допомагають у роботі з базою даних. Основні функції доступу до бази даних виконуються у групі класів модель представлення. Класи даної групи базуються на абстрактному класі BaseViewModel, для узагальнення та коректності роботи з базою даних. У кожному з класів даної групи реалізується потрібний набір методів для роботи з базою даних, прикладом є метод вибірки питань у вибраних курсах:

public void GetQuestionsByCoursesId(int[] ids)

{

var modules = database.GetAllWithChildren<Module>(m => ids.Contains(m.CourseId), true);

List<Question> q = new List<Question>();

foreach (var module in modules)

{

foreach (var theme in module.Themes)

{

q.AddRange(theme.Questions);

}

}

Entities = new ObservableCollection<Question>(q);

}

У даному методі виконується поступова вибірка питань з бази даних по полю Id таблиці курсів (Course). Тому для отримання необхідних записів, після запиту до бази даних відбувається циклічний перебір об’єктів та в результаті створюється об’єкт типу ObservableCollection з потрібним списком записів.

Для виконання основної з функцій додатку (проведення тестування), реалізований додатковий клас який також входить у групу модель представлення (ViewModel) та містить у собі набір методів та полів по роботі з списком вибраних для тестування питань, а саме:

* Відображення питання
* Перевірка відповідей усіх типів (багато вірних відповідей, одна вірна відповідь та відповідь введена з клавіатури)
* Перевірка на еквівалентність питання
* Підрахунок загальної кількості питань
* Підрахунок питань з правильною відповіддю
* Робота з таймером

Після закінчення тестування користувачу (студенту) відображається результат тестування, який буде записаний до локальної бази даних. Прикладом методу який виконує відображення та збереження результату є:

private void TestFinish()

{

timer.Stop();

MessageBox.Show(string.Format("{0} вірних, з {1} питань",qNotify.TrueQuestion, qNotify.NumberQuestion),"Результат", MessageBoxButton.OK);

testVM.AddTest(new Test(type, questionVM.GetNames(type, str.Split(' ').Select(int.Parse).ToArray()), qNotify.NumberQuestion, qNotify.TrueQuestion, DateTime.Now.ToShortTimeString(), DateTime.Now.ToShortDateString(), qNotify.Timer.ToString()));

NavigationService.Navigate(new Uri("/View/CourseView.xaml", UriKind.Relative));

}

В вище наданому прикладі наведено програмний код який виконується після завершення виконання тестування, виконує відображення результату створення об’єкту класу Test який зберігається у локальну базу даних та переходить на головний екран додатку. Також у додатку Г відображена реалізація класу групи ViewModel, QuestionNotify для роботи з екраном проходження тестування.

* + - 1. Програмна реалізація серверної частини

Як основу для серверної частини використовується бібліотека докер яка після встановлення в операційну систему Linux дає можливість налаштування та створення потрібної кількості віртуальних машин. Опис функціонування даної бібліотеки описується у підпункті 2.3. вибір інструментів для реалізації додатку. Під час процесу налаштування контейнерів Docker виконується налаштування та завантаження програмного забезпечення у кожен з них (додаток Д).

Реалізація серверної частини використовується Php Framework Symfony версії 3.2.8. Встановлення фреймворку у контейнер testing\_api\_php виконується за допомогою менеджера залежностей Composer (бібліотека для налаштування залежностей між компонентами у проекті) [21].

Для реалізації вище описаної функціональності даної підсистеми встановлено та налаштовано такі модулі:

* FOSUserBundle – забезпечує роботу з користувачем (можливості авторизації у системі) [22].
* FOSRestBundle – реалізує можливість генерації маршрутів для API та забезпечує роботу над об’єктами[23].
* NelmioApiDocBundle – реалізує генерацію специфікації по використанню маршрутів API [24].
* JMSSerializerBundle – надає можливість налаштування серіалізації об’єктів під час виконання запитів у формат json та навпаки [25].

Реалізація функціональності системи міститься у створеному по замовчуванню під модулі AppBundle.

Для надання можливості відображати необхідні екрани, сторінка результатів, сторінка зі списками (курсів, модулей, тем, питань), реалізований контролер та відповідні методи для кожної зі сторінок. Приклад програмного коду контролера та методів відображення сторінок:

class DefaultController extends Controller

{

/\*\*

\* @Route("/", name="homepage")

\*/

public function indexAction(Request $request)

{

return $this->render('AppBundle:pages:index.html.twig');

}

/\*\*

\* @Route("/list", name="list")

\*/

public function listAction(Request $request)

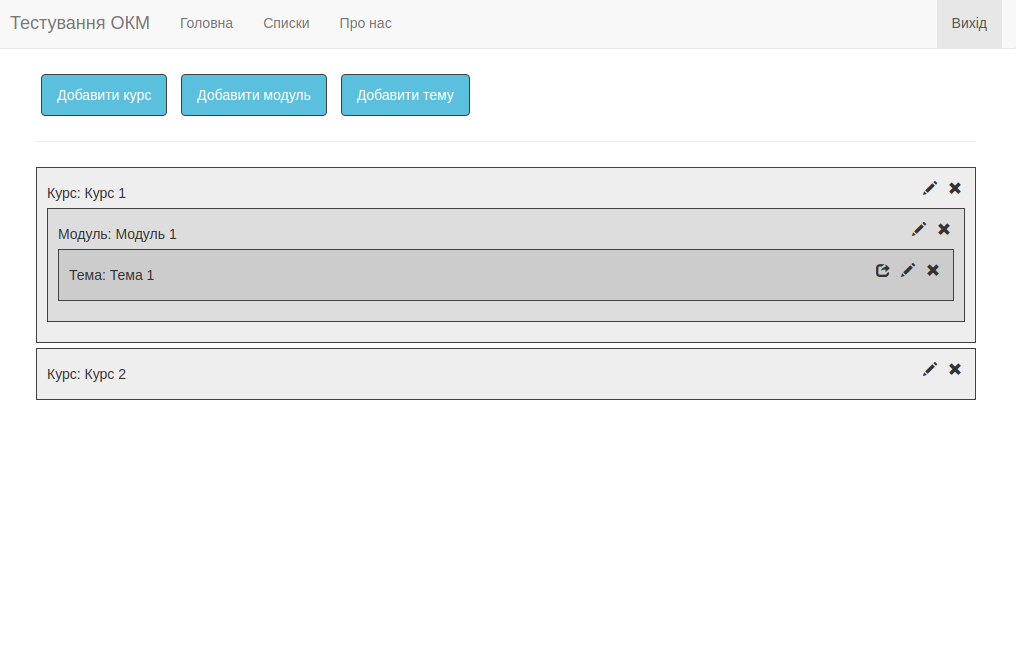
{

return $this->render('AppBundle:pages:list.html.twig');

}

}

У вище наданому прикладі програмного коду відображено метод створення маршрутів та прив’язка до файлу розмітки. У даному випадку використовується шаблонізатор Twig, який надає додаткові можливості при роботі з фреймворками.



* + 1. Приклад сторінки списків

На даній сторінці користувачу надається можливість перегляду та редагування курсів, модулів, тем, питань та відповідей.

Для генерації блоків на сторінці використовується бібліотека JQuery, яка з легкістю надає можливість керування елементами DOM структури сторінки.

Завантаження та усі дії на записами виконуються за допомогою Ajax запитів, які надають можливість передачі інформації між сторінкою та сервером без перезавантаження сторінки та динамічно виконувати потрібні дії над записами. Прикладом програмного коду роботи над записами є:

//завантаження даних

function updateArray() {

$.getJSON(url, function (data) {

generateList(data);

array = data;

});

}

//блок видалення

$(document).on('click', 'button.delete.course', function () {

var id = $(this).parent().attr('obj\_id');

var url = '/api/courses/' + id;

query(url, 'DELETE', function (e) {

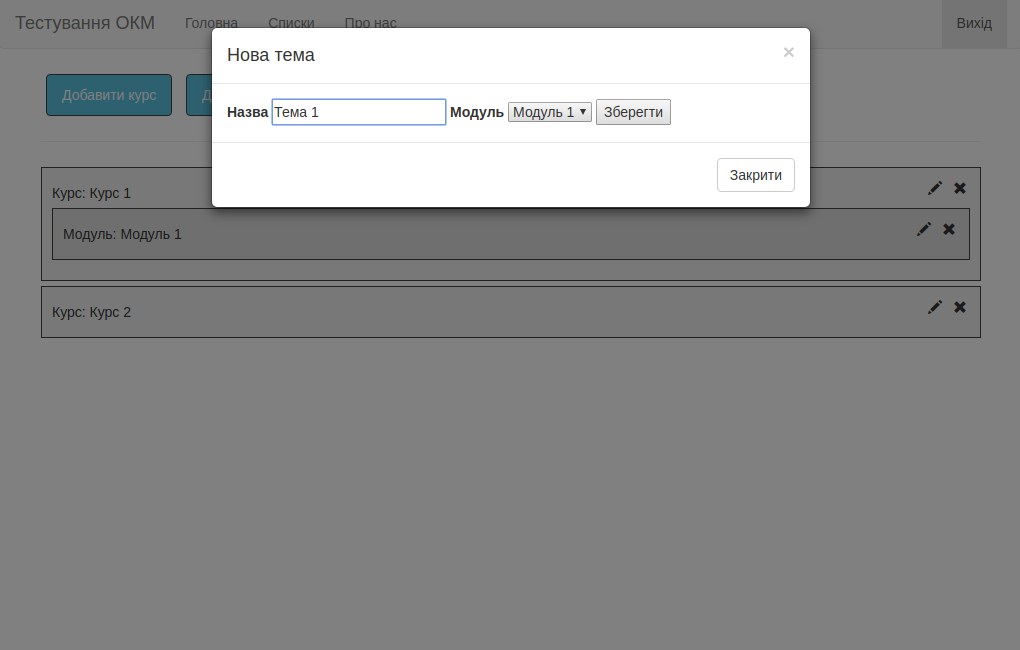
updateArray();

}, null)

});

У вище наданому прикладі програмного коду використовуються Ajax запити типів GET, POST, PUT та DELETE, які відповідно надають можливість отримати, записати, редагувати та видаляти записи з бази даних за допомогою API маршрутів.

Для виконання дій над записами використовуються модальні вікна. Вони є інтуїтивно зрозумілими та необхідними. Прикладом є модальне вікно створення теми, на якому користувачу надається можливість ввести назву та вибрати до якого модуля буде належати тема (рис. 4.2).



* + 1. Модальне вікно створення теми

Для створення та налаштування маршрутів API використовується модуль FOSRestBundle який надає можливість розширенням класу FOSRestController створювати маршрути для кожного об’єкту. Реалізація маршрутів відбувається за допомогою класів які у свою чергу наслідують клас AbstractApiController, у якому описуються методи доступу до бази даних (додаток Е).

У даній підсистемі використовується методи захисту маршрутів які описуються у файлі конфігурації доступів та методів авторизації security.yml. При переході на сторінки без пройденої авторизації система буде перенаправляти користувача на сторінку авторизації, яка дає можливість вести логін та пароль і авторизуватись.

Для перевірки прав доступу користувача на сторінках для відображення потрібних блоків використовується синтаксична конструкція шаблонізатора яка використовується на головній сторінці та виглядаю наступним чином:

{% if is\_granted('ROLE\_USER') %}

<div class="col-md-12">

<ul class="list">

</ul>

</div>

{% else %}

<div class="col-md-12">

<h1 class="text-center text-danger">Для перегляду увійдіть у систему!!!</h1>

</div>

{% endif %}

У представленому програмному коді виконується перевірна на наявність ролі ROLE\_USER у даного користувача, у результаті чого відображаються необхідні блоки.

Для вибірки потрібних записів з бази даних фреймворк Symfony використовує набір класів репозиторіїв які підключаються відповідно до кожного з класів моделі даних. Вони використовуються для опису методів які реалізуються запити до бази даних. Прикладом реалізації запиту є метод вибірки питань по полю theme\_id:

class QuestionRepository extends \Doctrine\ORM\EntityRepository

{

public function getQuestionByThemeId($id){

return $this->createQueryBuilder('q')

->where('q.theme = :theme\_id')

->setParameter('theme\_id', $id)

->getQuery()

->getResult();

}

}

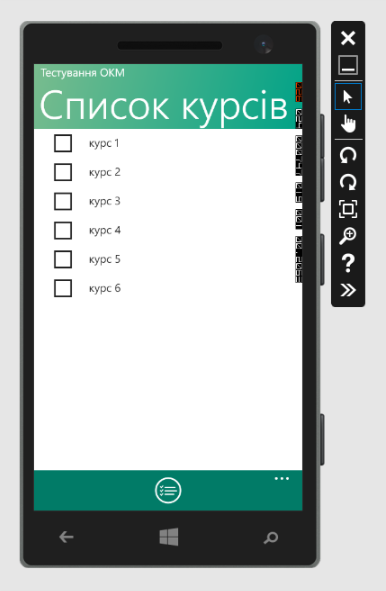
У даному прикладі реалізується метод запиту до бази даних з параметром $id, який передається у запит. Він використовується на сторінці списків під час відображення питань теми.

* 1. Тестування розробленої системи
     + 1. Тестування мобільного додатку

Тестування мобільного додатку виконується за допомогою емуляторів, які надають можливість перевірки додатку у різних умовах та на різних пристроях, які мають різну характеристику та можливості (додаток В).

Також тестування виконувалось на реальних пристроях які також мають різну характеристику. У даному випадку використовувались пристрої Nolia Lunia 525 та Microsoft Lumia 640, які в своїх характеристиках мають явну різницю у розширенні екрану, що дає можливість перевірити коректність роботи екранів та паралельно дає можливість тестування функціональності [26].

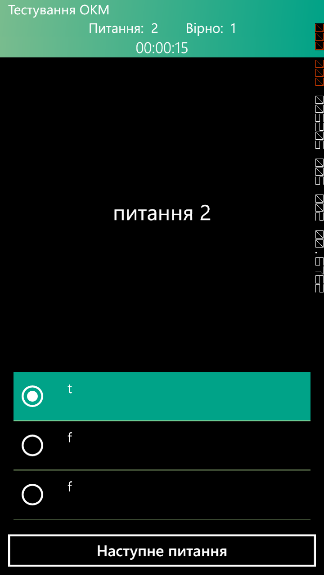
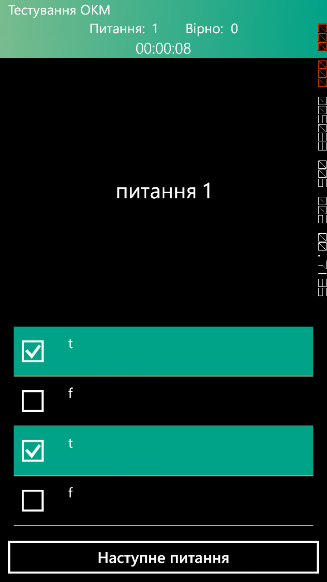
Прикладом тестування додатку на емуляторі є рис. 4.3.



* + 1. Приклад тестування додатку на емуляторі

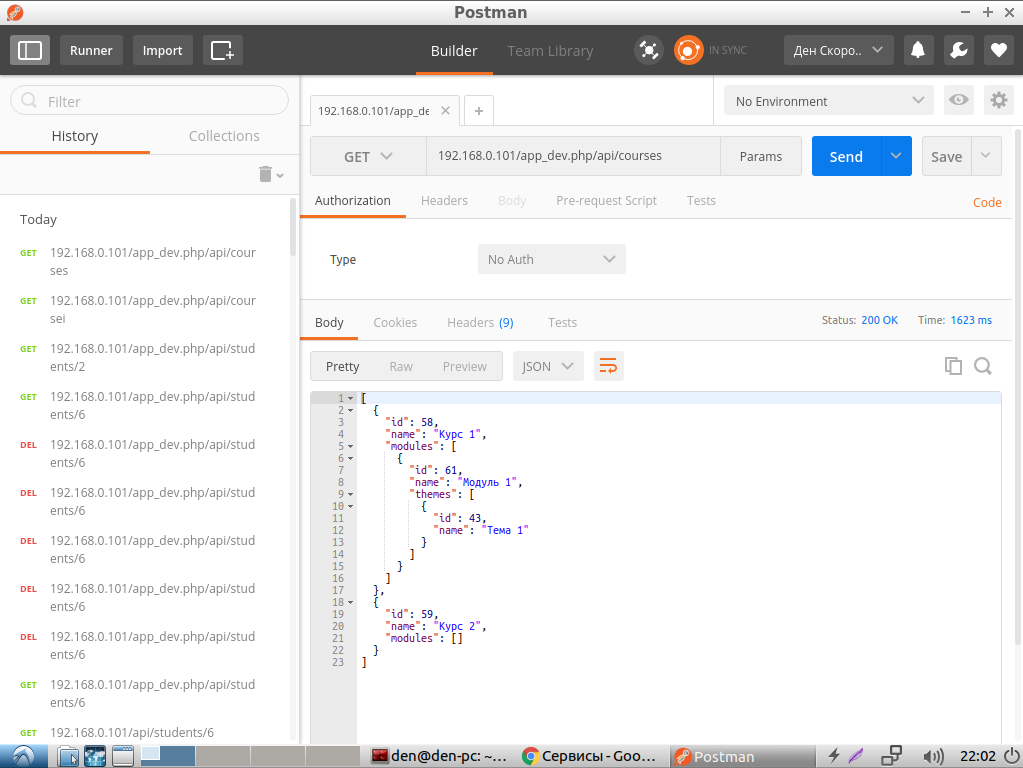
На рис. 4.3 зображено виконання мобільного додатку на емуляторі операційної системи Windows Phone 8.1. Під час виконання на емуляторах у правій частині екрану відображаються відомості про виконання додатку а саме, кількість кадрів за швидкість потоку кадрів, використана пам’ять, кількість опрацьованих пікселів та інші відомості про візуалізацію.

Приклад тестування мобільного додатку на пристрої Microsoft Lumia 640 зображено на рис. 4.4.

* + 1. Приклад тестування на пристрої Microsoft Lumia 640
       1. Тестування серверної частини

Оскільки серверна частина логічно поділяється ще на дві частини, їх тестування також проходить окремо. Тестування першої частини як API та систему маршрутизації проводиться за допомогою такого програмного забезпечення як Postman [27].



* + 1. Тестування API за допомогою програмного забезпечення Postman

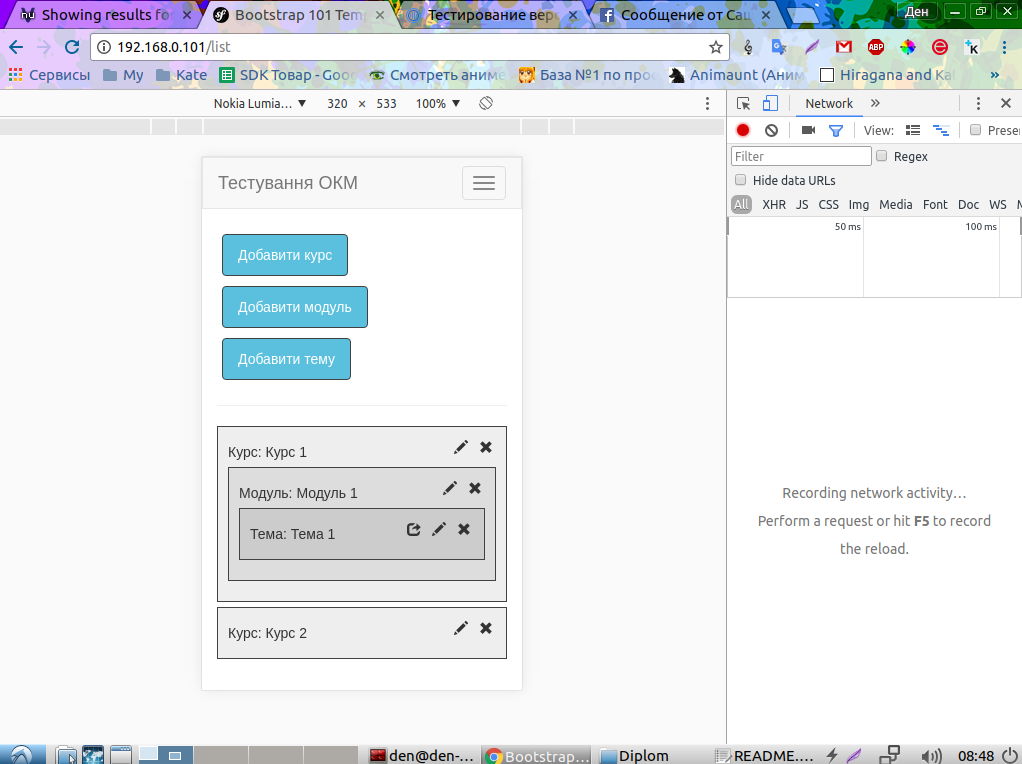
За допомогою нього виконується перевірка запитів до маршрутів API, тобто усіх маршрутів як починаються з «/api». У результаті проведення кожного запиту відбувається перевірка статус коду (200, 400, 401, 405 тощо), перевірка часу обробки запиту та сама відповідь яка приходить від сервера (рис. 4.5).

На рис. 4.5 відображено виконання GET запиту на отримання списку курсів. В результаті отримано статус код 200, час запиту 1623 мс та у відповідь текст у форматі Json, що відповідає функціональним вимогам.

Тестування другою логічної частини, тобто зовнішнього вигляду (верстки) перевіряє коректність роботи у середовищах різного типу. Процес тестування складається з наступних етапів:

* Тестування на коректність роботи у різних браузерах.
* Перевірка масштабування на екрані з розширенням 1024 пікселя та більших.
* Перевірка валідності верстки.
* Перевірка коректної роботи при використанні реального тексту.
* Перевірка працездатності сторінок при виключеному JavaScript.

Прикладом перевірки масштабування є рис. 4.6.



* + 1. Приклад тестування масштабності верстки

На рис. 4.6 зображено тестування верстки на масштабність за допомогою можливості браузера Google Chrome емулювати роботу веб сторінки на вказаному пристрої, у даному випадку емулюється коректність відображення веб сторінки на пристрої Nokia Lumia 520.

1. економічна частина
   1. Економічна характеристика проектного рішення (програмного продукту)

Метою даного проекту є надання допомоги студентам у процесі підготовки до іспитів, контрольних, та інших тестових робіт та невід’ємній її частині, самоперевірці засвоєних знань.

Доцільність розроблення даного проекту полягає у наданні допомоги студентам які використовують мобільні пристрої на базі операційної системи Windows Phone 8.1, адже розглядаючи аналоги програмного продукту усі з них були розроблені для операційних систем Android та IOS.

Тому спираючись на це можна з впевненістю можна сказати, що попит на користування даного проекту буде не малий.

Проект на даному етапі розробки не має багато шансів окупитись, але майбутні версії можуть бути платні. Це дає змогу запропонувати функціональні можливості проекту великій кількості користувачів, та у майбутньому з розширенням функціоналу збільшувати дохід виручений за користування.

* 1. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення проектного рішення (програмного продукту)

На сьогоднішній день розробка мобільних додатків є дуже популярною. Завантаження різноманітних додатків уже не є дивиною для користувачів, адже кожна з мобільних операційних систем надає доступ до інтернет магазину додатків у відповідності з операційною системою.

За оцінками міжнародної компанії AppAnnie, ринок мобільних додатків буде стрімко зростати. За 2015 рік дохід мобільних додатків сягає $ 70 млрд та буде зростати до $ 189 млрд 2020 року. Користувачі будуть у двічі більше часу витрачати на мобільні додатки та мобільні ігри будуть приносити більше половини усіх доходів [28].

Для характеристики аналогічного продукту був вибраний мобільний додаток «Тести Крок». Даний додаток розроблений для операційної системи Android та орієнтований на студентів медичних навчальних закладів та його мета, допомогти підготуватись до складання таких іспитів: крок м, крок б, крок 1-3. На ринку мобільний додаток є безкоштовним, для завантаження потрібно всього лиш перейти у Play Маркет та ввести у полі пошуку назву додатку. Подібних додатків на ринку є багато але їх функціонал та можливості є різними тому перш ніж завантажувати потрібно уважно переглянути опис та відгуки.

Також необхідно запевнити, що процес інтеграції продукту на ринок не є безкоштовним. Для реєстрування компанії або розробника особисто необхідно заплатити одноразовий внесок у розмірі 99$ та 19$ відповідно (інформація зазначена для магазину Windows Store) [29].

Основними споживачами даного проекту та його аналогів є користувачі віком від 17 до 30 років, тобто в загальному це студенти. Потребами студентів зокрема перед іспитами є підготовка та самоперевірка рівня знань. Звісно розширюючи функціонал студенту надається можливість не тільки підготуватись до заліку але і засвоювати матеріал під час навчального процесу, але це залежить від конкретного додатку.

Основними конкурентами у даному напрямку ринку є розробки різноманітних компаній які навіть при першому вивантажені додатку в інтернет магазин встановлюють ціну на програмний продукт. Функціонал таких додатків є більш розширеним але не кожен студент має можливість витрати коштів на програмне забезпечення.

Одже аналізуючи ринок, аналоги, основних споживачів та конкурентів можна дійти висновку що розробка даної бакалаврської роботи є необхідною з можливістю подальшого оновлення та розширення функціоналу. Основними характеристиками є безкоштовне надання допомоги під час підготовки до іспитів, потенційними користувачами додатку є студенти вищих навчальних закладів.

* 1. Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ

Для визначення загальної оцінки впливу середовищ необхідно сформувати таблицю факторів впливу зовнішнього та внутрішнього середовищ.

**Результати експертного оцінювання впливу факторів зовнішнього та внутрішнього середовищ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Фактори** | **Середня експертна оцінка, бали** | **Середня вагомість факторів** | **Зважений рівень впливу, бали** |
| 1 | 2 | 3 | 4 |
| *Фактори зовнішнього середовища* |  | | |
| Споживачі | +5 | 0,11 | 0,55 |
| Постачальники | 0 | 0,1 | 0,4 |
| Конкуренти | +3 | 0,1 | 0,2 |
| Державні органи влади | -3 | 0,05 | -0,2 |
| Інфраструктура | -2 | 0,06 | -0,12 |
| Законодавчі акти | -3 | 0,1 | -0,3 |
| Профспілки, партії та інші громадські організації | +1 | 0,05 | 0,05 |
| Система економічних відносин в державі | +2 | 0,06 | 0,18 |
| Організації-сусіди | 0 | 0,01 | 0 |
| Міжнародні події | -4 | 0,01 | -0,04 |
| Міжнародне оточення | +2 | 0,03 | 0,06 |
| Науково-технічний прогрес | +4 | 0,07 | 0,28 |
| Політичні обставини | +2 | 0,06 | 0,12 |
| Соціально-культурні обставини | -1 | 0,05 | -0,05 |
| Рівень техніки та технологій | +4 | 0,04 | 0,16 |
| Особливості міжнародних економічних відносин | +3 | 0,02 | 0,06 |
| Стан економіки | -3 | 0,08 | -0,24 |
| Загальна сума |  | 1 |  |
| *Фактори внутрішнього середовища* |  | | |
| Цілі | +5 | 0,11 | 0,55 |
| Структура | +3 | 0,16 | 0,48 |
| Завдання | +4 | 0,07 | 0,28 |
| Технологія | +5 | 0,2 | 0,6 |
| Працівники | +5 | 0,21 | 1,05 |
| Ресурси | +3 | 0,25 | 0,75 |
| Загальна сума |  | 1 |  |

Виходячи з результатів обрахунків зображених у табл. 1, на діяльність організації найбільш позитивно впливають такі фактори зовнішнього середовища, як споживачі, конкуренти, система економічних відносин у державі, науково технічний прогрес, особливості міжнародних економічних відносин. Потрібно зауважити, що найбільш негативно на діяльність організації впливають такі фактори, як державні органи влади, інфраструктура, законодавці акти, соціально-культурні обставини.

Також у межах табл. 1 зображено вплив факторів внутрішнього середовища. В загальному усі фактори впливають позитивно але необхідно зазначити, що найбільш розвинутими є вплив працівників, технологій та поставлених цілей. Менш впливовими на діяльність організації є такі фактори, як неструктурованість завдань та ресурси, які є необхідними.

* 1. Формування стратегічних альтернатив

Базуючись на характеристиці даного проекту (пункт 5.1), та інформаційному забезпеченні (пункт 5.2), можна запевнитись що даний напрямок не є новим та розробка аналогів відбувається стрімко. Тому за критеріями першої групи стратегічних альтернатив, одразу можна обрати існуючий продукт. Однак вибір між критеріями додаткових послуг є настільки швидким.

Якщо повернутись до огляду аналогів можна зазначити, що варіації у функціональних можливостях є дуже різні. Тому більш доцільним буде вибір критерії наявних послуг. Адже пропозиція продукту з більшою кількість послуг задовольняє потреби більш широкої аудиторії.

Вибір стратегічних альтернатив за другою групою (ринок та продукт) відбувається на основі стану ринку та його перспектив (пункт 5.2). ринок мобільних додатків у напрямку навчання є достатньо молодим, адже його розвиток залежить від розвитку та впровадження технологій у побут. Виходячи з цього можна сказати, що ринок є більш новий але продукт на ньому є достатньо існуючий та розвинений, тому характеристику продукту однозначно визначити не можна. У даному випадку буде доцільнішим вибір продукту як новий та вибір альтернативної стратегії розвитку, диверсифікація тобто розширення функціоналу та послуг для виходу на нові сфери бізнесу.

* 1. Бюджетування

За допомогою комплексного обґрунтування витрат (бюджетування) буде проаналізовано рентабельність виробництва, визначено собівартість виробництва та доцільність вибору однієї зі стратегій.

**Бюджет витрат матеріалів та комплектуючих виробів**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва матеріалів та комплектуючих** | **Марка, тип, модель** | **Фактична кількість, шт.** | **Ціна за одиницю, грн.** | **Разом, грн.** |
| Системний блок | Asus | 3 | 3560 | 10680 |
| Монітор | Philips 226V | 3 | 1780 | 5340 |
| Клавіатура | Genius | 3 | 165 | 495 |
| Мишка комп’ютерна | Sven RX-370 | 3 | 135 | 405 |
| Мережевий фільтр | - | 1 | 85 | 85 |
| Мобільний телефон | Nokia Lumia 525 | 1 | 1450 | 1450 |
| Кабель USB | - | 1 | 35 | 35 |
| Разом: |  | 15 | 7210 | 18490 |

**Бюджет витрат на утримання та експлуатацію устаткування**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Назва об’єкта устаткування, марка, тип, модель** | **Первісна вартість, грн** | **Термін експлуатації, років** | **Річна сума амортизаційних відрахувань, грн** | **Час використання, дні** | **Амортизаційні відрахування за період використання, грн** |
| Системний блок | 3560 | 4 | 890 | 1008 | 3,5 |
| Монітор | 1780 | 4 | 445 | 1008 | 1,8 |
| Клавіатура | 165 | 3 | 55 | 756 | 0,2 |
| Мишка комп’ютерна | 135 | 3 | 45 | 756 | 0,2 |
| Мережевий фільтр | 85 | 5 | 17 | 1260 | 0 |
| Мобільний телефон | 1450 | 3 | 484 | 756 | 1,9 |
| Кабель USB | 35 | 1 | 35 | 252 | 0,1 |
| Разом | 7210 | 26 | 1971 | 5796 | 7,5 |

**Бюджет витрат на оплату праці**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Посада,**  **спеціальність** | **Кількість працівників, осіб** | **Час роботи, дні** | **Денна заробітна плата працівників, грн.** | **Сума витрат на оплату праці, грн.** |
| *Основна заробітна плата* | | | | |
| Менеджер | 1 | 35 | 160 | 5600 |
| Дизайнер | 1 | 15 | 190 | 2850 |
| Розробник | 1 | 42 | 250 | 10500 |
| Разом: | 3 | 92 | 600 | 18950 |
| *Додаткова заробітна плата* | | | | |
| Розробник | 1 | 10 | 150 | 1500 |
| Разом: | 1 | 10 | 150 | 1500 |

**Бюджет обов’язкових відрахувань та податків**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Посада,**  **спеціальність** | **Сума**  **основної заробітної плати** | **Сума додаткової заробітної плати** | **Разом витрат на оплату праці** | **Сума єдиного внеску на соціальне страхування\*, грн.** | **Сума утримань (податку на доходи фізичних осіб, військового збору\*\*), грн.** |
| Менеджер | 3000 | 200 | 3200 | 704 | 576 |
| Дизайнер | 2500 | 0 | 2500 | 550 | 450 |
| Розробник | 3700 | 500 | 4200 | 924 | 756 |
| Разом: | 9200 | 700 | 9900 | 2178 | 1782 |

**Бюджет загальновиробничих витрат**

|  |  |
| --- | --- |
| **Статті витрат** | **Сума, грн.** |
| *Змінні загальновиробничі витрати, у т.ч.:* |  |
| - заробітна плата допоміжного персоналу; | 0 |
| - витрати на МШП; | 300 |
| - витрати на електроенергію; | 968,4 |
| - витрати на ремонт; | 500 |
| - інші змінні витрати; | 0 |
| Разом змінних витрат: | 1768,4 |
| *Постійні загальновиробничі витрати, у т.ч.:* |  |
| *-* заробітна плата допоміжного персоналу; | 0 |
| - комунальні послуги; | 3150 |
| - витрати на оренду; | 0 |
| - витрати на ремонт; | 500 |
| - інші постійні витрати; | 250 |
| Разом постійних витрат: | 3900 |
| *Разом загальновиробничих витрат:* | 6500 |

**Бюджет адміністративних витрат та витрат на збут**

|  |  |
| --- | --- |
| **Статті витрат** | **Сума, грн.** |
| 1 | 2 |
| *Адміністративні витрати, у т.ч.:* |  |
| - заробітна плата адміністративного персоналу; | 0 |
| - витрати на МШП; | 500 |
| - витрати на відрядження; | 0 |
| - витрати на ремонт; | 500 |
| - витрати на паливно-мастильні матеріали; | 0 |
| - витрати на сплату податків і зборів; | 3960 |
| - знос адміністративного обладнання; | 200 |
| - інші адміністративні витрати; | 0 |
| Разом адміністративних витрат: | 5160 |
| *Витрати на збут, у т.ч.:* |  |
| - заробітна плата менеджерів зі збуту; | 3000 |
| - витрати на гарантійний ремонт; | 0 |
| - витрати на відрядження; | 0 |
| - витрати на гарантійне обслуговування; | 0 |
| - витрати на налагодження і експлуатацію; | 0 |
| - витрати на паливо-мастильні матеріали; | 0 |
| - витрати на рекламу; | 850 |
| - інші витрати на збут; | 0 |
| Разом витрат на збут: | 3850 |

**Зведений кошторис витрат на розробку проектного рішення (продукту)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Статті витрат** | **Одиниці виміру** | **Фактична кількість, шт.** | **Ціна одиниці, грн.** | **Разом, грн.** |
| Сировина і матеріали | - | 0 | 0 | 0 |
| Купівельні напівфабрикати та комплектуючі вироби | - | 0 | 0 | 0 |
| Зворотні відходи (вираховуються) | - | 0 | 0 | 0 |
| Паливо та електроенергія на технологічні цілі | кв/год | 1076 | 0,9 | 968,4 |
| Основна заробітна плата | людина | 3 | 3300 | 9900 |
| Додаткова заробітна плата | людина | 2 | 500 | 1000 |
| Відрахування на соціальне страхування | людина | 3 | 982 | 2948 |
| Витрати на утримання й експлуатацію устаткування | грн | 1 | 250 | 250 |
| Загальновиробничі витрати, у т.ч.: |  |  |  |  |
| - змінні; | грн | 3 | - | 1768,4 |
| - постійні; | грн | 3 | - | 6500 |
| *Разом виробничих витрат:* |  | 6 | - | 8268,4 |
| Адміністративні витрати | грн | 4 | - | 7482 |
| Витрати на збут | грн | 1 | - | 850 |
| Інші операційні витрати | - | 0 | 0 | 0 |
| *Разом виробничих і операційних витрат:* |  |  |  | 8268,4 |

**Бюджет фінансових результатів**

|  |  |
| --- | --- |
| **Показники** | **Сума, грн.** |
| 1 | 2 |
| Дохід від реалізації продукції | 47250 |
| Податок на додану вартість\*\*\* | 9450 |
| Чистий дохід від реалізації продукції | 37800 |
| Собівартість реалізованої продукції | 31667 |
| Валовий прибуток | 6133 |
| Операційні витрати: |  |
| - адміністративні витрати: | 7482 |
| - витрати на збут; | 850 |
| - інші операційні витрати; | 0 |
| Фінансовий результат від операційної діяльності |  |
| Податок на прибуток \*\*\*\* | 1104 |
| Чистий прибуток (збиток) | 5029 |

Одже, після обрахунків можна зазначити що розробка даного проекту є рентабельною. Вибір стратегій диверсифікації є доцільним, адже наступні версії проекту з розширеним функціоналом будуть приносити дохід.

* 1. Вибір стратегії

Вибір стратегії розвитку підприємства є одне з головних питань на ранніх стадіях розвитку. Базовою інформацією для вибору є аналітичні данні ринку аналогів, потреби користувачів, зовнішні та внутрішні середовища та інше.

У даному випадку аналіз ринку чітко відображає велику кількість аналогів але навіть половина не є конкурентно спроможніми. Тобто розвиваючи даних напрямок та необхідні для цього технології можна досягнути лідерства. Цільовою аудиторією та її потребою для додатків даного напрямку є студенти яким потрібна допомога у підготовці до різнотипних іспитів, контрольних робіт, тестів тощо.

Найбільш вагомими є фактори впливу зовнішнього та внутрішнього середовища які були проаналізовані у пункті 5.3. На основі цього більш впливовими факторами є споживачі, конкуренти, рівень техніки та технологій та менш впливовими міжнародні відносини, стан економіки та державні органи влади.

На основі інформації попередні пунктів можна сказати, що вибір стратегії лідерства у ніші буде найдоцільнішим.

Адже розвиваючи функціонал та технології розробки подібних додатків, забезпечать стабільний дохід та зменшать ризики до мінімуму, що є основним у виборі стратегії розвитку. Досягнувши необхідного рівня та зайнявши місця лідера на ринку, процес визначення стратегії може бути повторено.

Висновки

Одже метою даного проекту є надати студентам допомогу під час підготовки до різнотипних іспитів, контрольних та інших тестових робіт за допомогою самоперевірки та проходження тестування.

Для вирішення поставленого завдання було розглянуто можливий ринок збуту, аналоги програмного забезпечення їх можливості та недоліки, потреби користувача та інструментальні засоби за допомогою яких відбувалась проектування та реалізація проекту.

На основі аналізу вище описаних джерел за основу була вибрана платформа .Net та пристрої на основі операційної системи Windows Phone 8.1. Для реалізація програмного коду мобільного додатку основною була вибрана мова програмування C# та мова розмітки XAML для реалізації зовнішнього вигляду.

Для написання серверної частини використовувалась мова Php та шаблонізатор Twig який у свою чергу генерує представлення даних мовою розмітки Html. Також додатково використані мови JavaScript з бібліотекою Jquery та каскадні таблиці стилів Css для оформлення.

Середовищем реалізації функціональності мобільного додатку вибрано Visual Studio 2015 та пакет розробника Windows Phone SDK 8.1. У можливості даного пакету входить набір компіляторів та інтерпретаторів, набір віртуальних машин мобільних пристроїв та допоміжні інструменти для роботи з репозиторієм та інше.

Для серверної частини середовищем було вибрано IDE PhpStorm 2017 та бібліотеку Docker яка надаючи набір віртуальних машин, однією з яких є testing\_api\_nginx , реалізувала середовище функціонування. Фреймворком для реалізації функціональності серверної частини став Symfony 3.2.8 та набір вбудованих та додаткових модулів.

У результаті реалізації даного проекту було створено мобільний додаток та серверну частину API з візуальним блоком. Які надають можливість проходження тестування з збереженням результатів та їх вивантаженням за допомогою API. Серверна частина реалізує функціональність API та візуальний блок для роботи з базою даних та перегляду результатів тестування.

Список літератури

1. Центру тестування при МОЗ України [Електронний ресурс]. – Режим доступу: http://testcentr.org.ua/index.php/about-us/menu-contacts.html.
2. Сучасні методи оцінювання знань, комп’ютерне тестування [Електронний ресурс]. – Режим доступу: http://confesp.fl.kpi.ua/node/1021.
3. Тестування як ефективний метод перевірки професійної компетентності студентів [Електронний ресурс]. ― Режим доступу: http://osvita.ua/school/lessons\_summary/edu\_technology/15024/
4. Методичні рекомендації до підготовки та складання тестових завдань для викладачів, аспірантів вищих навчальних закладів [Електронний ресурс]. – Режим доступу: http://dl.lnu.edu.ua/metod/metod.htm.
5. Тести як одна із форм поточного контролю [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/fiznet11/metodicna-skarbnicka-1/metodika-vikladanna-1/testi-ak-odna-iz-form-potocnogo-kontrolu.
6. Мобільний додаток «TestTerra (ЗНО)» [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.testterra.main&hl=uk.
7. Мобільний додаток «Тести крок» [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.udelphi.testykrok.
8. Мобільний додаток «Крок помічник» [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=sezyakot.com.stepcrib.
9. Архів Windows SDK и емуляторів [Електронний ресурс]. – Режим доступу: https://developer.microsoft.com/ru-ru/windows/downloads/sdk-archive.
10. SQLitePCLRaw.lib.e\_sqlite3.v110\_wp80 1.1.5 NuGet Package [Electronic resource]. ― Web page: https://www.nuget.org/packages/SQLitePCLRaw.lib.e\_sqlite3.v110\_wp80.
11. SQLitePCLRaw.provider.e\_sqlite3.wp80 1.1.5 NuGet Package [Electronic resource]. ― Web page: https://www.nuget.org/packages/SQLitePCLRaw.provider.e\_sqlite3.wp80/.
12. Windows Phone Toolkit [Electronic resource]. ― Web page: https://www.nuget.org/packages/WPtoolkit/.
13. Docker [Electronic resource]. ― Web page: https://www.docker.com/.
14. Дудзяний І. М. Обєктно-орієнтоване моделювання програмних систем [Електронний ресурс]. ― Режим доступу: http://old.ami.lnu.edu.ua/books/AMI/OOM.pdf.
15. Паттерн MVVM [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/wpf/22.1.php.
16. Чернецки К., Айзенекер У. Порождающее программирование. Методы, инструменты, применение. — Издательский дом «Питер», 2005. — 730 с.
17. Общие сведения о языке XAML (WPF) [Електронний ресурс]. – Режим доступу: https://msdn.microsoft.com/ru-ru/library/ms752059(v=vs.110).aspx.
18. Quickstart: Creating a user interface with XAML for Windows Phone 8 [Electronic resource]. ― Web page: https://msdn.microsoft.com/ru-ru/library/windows/apps/jj207025(v=vs.105).aspx.
19. Программируем Windows phone 7, Чарльз Петзольд [Електронний ресурс]. – Режим доступу: http://rusdpe.blob.core.windows.net/downloads/Programming\_Windows\_Phone\_7\_ru.pdf.
20. Microsoft developing, How to create your first app for Windows Phone 8 [Electronic resource]. ― Web page: https://msdn.microsoft.com/ru-ru/library/windows/apps/ff402526(v=vs.105).aspx.
21. Symfony, Installing & Setting up the Symfony Framework [Electronic resource]. ― Web page: http://symfony.com/doc/current/setup.html
22. Getting Started With FOSUserBundle [Electonic resource]. ― Web page: http://symfony.com/doc/current/bundles/FOSUserBundle/index.html
23. Getting Started With FOSRestBundle [Electronic resource]. ― Web page: http://symfony.com/doc/current/bundles/FOSRestBundle/index.html
24. NelmioApiDocBundle https://symfony.com/doc/current/bundles/NelmioApiDocBundle/index.html
25. JMSSerializerBundle http://jmsyst.com/bundles/JMSSerializerBundle
26. Microsoft developing, App performance considerations for Windows Phone 8 [Electronic resource]. ― Web page: https://msdn.microsoft.com/en-us/library/windows/apps/ff967560%28v=vs.105%29.aspx?f=255&MSPPError=-2147217396.
27. Postman [Electronic resource]. ― Web page: https://www.getpostman.com/
28. Ринку мобільних додатків прогнозують вражаюче зростання - на 270% до 2020 року [Електронний ресурс]. ― Режим доступу: http://osvita.mediasapiens.ua/web/IT\_companies/rinku\_mobilnikh\_dodatkiv\_prognozuyut\_vrazhayuche\_zrostannya\_na\_270\_do\_2020\_roku/.
29. Microsoft developing, register as an app developer [Electronic resource]. ― Web page: https://developer.microsoft.com/en-us/store/register.

Додаткові UML діаграми

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Виконав |  | Підпис | Дата | Діаграма кооперації (collaboration diagram) | Сторінка |
| Перевірив |  | Підпис | Дата | 1 з 2 |
| ІДН кафедра ПЗ ПІх-41з | | | |
|  | | | | | |
| Виконав |  | Підпис | Дата | Діаграма компонентів | Сторінка |
| Перевірив |  | Підпис | Дата | 2 з 2 |
| ІДН кафедра ПЗ ПІх-41з | | | |

Інструкція користувача мобільного додатку для тестування знань студентів з дисципліни «Організація комп’ютерних мереж»

**Компоненти ПЗ**

Мобільний додаток для тестування знань студентів розроблений на мові програмування C# 6.0 у середовищі Visual Studio .NET 2015 і може експлуатуватись на пристроях з операційною системою Windows Phone 8.1. також була розроблена підсистема (серверна частина API) для додаткового функціоналу.

Для коректної роботи проекту необхідний мобільний пристрій на основі операційної системи Windows Phone 8.1 з довільною характеристикою.

Для роботи серверної частини необхідна користувацька машина з процесором не менше 1.5 GHz, оперативною пам’ять не менше 512 Mb. Для експлуатації серверної частини під керуванням операційної системи Linux, необхідно мати в наявності встановлену бібліотеку Docker та допоміжну Docker-compose.

**Встановлення ПЗ**

Для роботи мобільного додатку необхідно встановити Visual Studio 2015. Після чого необхідно відкрити файли підсистеми у Visual Studio 2015 та обравши цільовий пристрій Device (підключений до машини пристрій за допомогою USB кабелю) та запустити на виконання. Надалі додаток буде викладено у вільний доступ у інтернет-магазин для полегшеного встановлення.

Серверну частину необхідні встановити за допомогою терміналу, перейшовши в підкаталог серверної частини /app/docker необхідно виконати команду docker-compose up -d та після автоматичного налаштування віртуальних машин виконати команду docker exec -it testing\_api\_php bash яка дасть можливість керування віртуальної машиною де розміщені файли.

Після попередньо вказаних дії необхідно встановити файли фреймворку за допомогою команди composer install, після необхідно виконати команду chmod -R 777 var/cache var/logs, що наддасть доступ системі для запису кешу та лог файлів які є обов’язково необхідними.

Результат встановлення серверної частини можна переглянути у браузері перейшовши по адресу попередньо визначеному у терміналі виконаною командою docker inspect testing\_api\_nginx.

**Базові функції ПЗ**

Під час першого запуску мобільного додатку необхідним є підключення до мережі інтернет для початкового завантаження бази даних з набором даних для виконання локального тестування. Надалі користувач може локально проходити тестування зі збереженням результатів у локальну базу даних. При бажанні користувач може переглядати результати з можливістю сортування по даті або типу тестування.

Також при необхідності користувач може перевіряти базу даних на оновлення та завантажувати останню версію.

Після переходу по вказаній адресі користувача буде направлено на сторінку авторизації де необхідно ввести логін та пароль. Надалі користувачу буде надано можливість переглянути результати вивантажені користувачами мобільного додатку. Також користувач має можливість для переходу на сторінку списків де відображається списки курсів, модулів, тем та інші. На даній сторінці користувач може виконувати необхідні операції з записами.

Звіт про тестування мобільного додатку для тестування знань студентів з дисципліни «Організація комп’ютерних мереж»

Вступ

Для перевірки коректності роботи будь якої системи, у процесі та вкінці розробки виконується тестування. Види тестування вибираються в залежності від типу системи, платформи для якої вона розроблена та особливостей самої системи.

Розробка тестів

У даному випадку основою проекту є мобільний додаток та для його тестування було вибрано такий метод тестування як функціональне тестування. Функціональне тестування є одним з видів незалежного тестування програмного забезпечення, яке ціллю якого є перевірка реалізованих функціональних вимог.

Функціональні тестові випадки

Під час проведення тестування мобільного додатку було створено 22 тестових випадки, які перевіряють найважливіші функції програми, та у них використано 26 випадкових наборів даних.У таблиці представлено розподіл функціональних тестових випадків та випадкових наборів даних для розроблених випадків (варіантів використання).

*Таблиця В.1*

**Результати проведення функціонального тестування**

|  |  |  |
| --- | --- | --- |
| **Варіанти використання** | **Тестові випадки** | **Тестові дані** |
| Авторизація користувача | 3 | 5 |
| Вибір курсів, модулів або тем | 6 | 6 |
| Проходження тестування | 9 | 12 |
| Перегляд результатів тестування | 1 | 1 |
| Вивантаження вибраних результатів тестування | 3 | 2 |
| **Загалом** | **22** | **26** |

Результати тестування

Після виконання тестових випадків (варіантів використання), серйозних недоліків або помилок не виявлено, 19 з 22 функціональних тестів були проведені успішно (24 з 26 наборів тестових даних пройшли). Одже функціональне тестування пройшло успішно. Всі дрібні помилки та недоліки були виправлені під час тестування, та були протестовані повторно. Коректність роботи відповідає функціональним вимогам.

**Критерії успіх/провал проекту:**

* Усі заплановані тестові випадки розроблено;
* Покриття тестами функціональних вимог не менше 85%;
* Успішне виконання 88% випадків використання;
* Усі заплановані функціональні тести виконані;

Висновки тестування

Одже на протязі процесу тестування мобільного додатку уразливих місць не знайдемо та серйозні дефекти не виявлено. Дрібні помилки були одразу виправлені. Під час тестування виконано перевірку можливих випадків дій користувача та реакція системи на них. У результаті функціональність виправлена повністю та помилок не містить.

Програмний код мобільного додатку (реалізація класу QuestionNotify)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

using Testing\_3.CompareString;

using Testing\_3.Model;

namespace Testing\_3

{

class QuestionNotify : INotifyPropertyChanged

{

int trueQuestion = 0;

public int TrueQuestion

{

set

{

trueQuestion = value;

NotifyPropertyChanged();

}

get

{

return trueQuestion;

}

}

int numberQuestion = 0;

public int NumberQuestion

{

set

{

numberQuestion = value;

NotifyPropertyChanged();

}

get

{

return numberQuestion;

}

}

int index = 0;

Question[] questions;

Question curentQuestion;

TimeSpan timer;

public TimeSpan Timer

{

set

{

timer = value;

}

get

{

return timer;

}

}

public Question CurentQuestion

{

set

{

if (!EquivalentQuestion.Contains(value))

{

curentQuestion = value;

EquivalentQuestion.AddRange(CurentQuestion.EquivalentQuestion);

NotifyPropertyChanged();

}

else

{

SkipQuestion();

}

}

get

{

return curentQuestion;

}

}

List<Question> EquivalentQuestion;

public QuestionNotify(Question[] questions)

{

EquivalentQuestion = new List<Question>();

this.questions = questions;

CurentQuestion = questions[index];

NumberQuestion++;

index++;

timer = new TimeSpan(0,0,0);

}

public bool NextQuestion(string textAnswer)

{

if (CurentQuestion.Answers.Count > 0)

{

var i = ScoreSharp.score(CurentQuestion.Answers[0].Text, textAnswer);

if (i > 0.7)

{

TrueQuestion++;

}

}

if (index++ < questions.Count())

{

CurentQuestion = questions[index++];

NumberQuestion++;

return true;

}

else

{

return false;

}

}

public bool NextQuestion(Answer answer)

{

foreach (var ans in CurentQuestion.Answers)

{

if (ans.Corect && ans.Id == answer.Id)

{

TrueQuestion++;

}

}

if (NumberQuestion < questions.Count() - 1)

{

CurentQuestion = questions[index++];

NumberQuestion++;

return true;

}

else

{

return false;

}

}

public bool NextQuestion(Answer[] answers)

{

var x = CurentQuestion.Answers.Where(a => a.Corect == true).Select(a => a.Id).ToArray();

var y = answers.Select(a => a.Id).ToArray();

if (x.SequenceEqual(y))

{

TrueQuestion++;

}

if (NumberQuestion < questions.Count() - 1)

{

CurentQuestion = questions[index++];

NumberQuestion++;

return true;

}

else

{

return false;

}

}

public void SkipQuestion()

{

CurentQuestion = questions[index++];

}

public event PropertyChangedEventHandler PropertyChanged;

private void NotifyPropertyChanged([CallerMemberName] string propertyName = "")

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

public TimeSpan AddTimer(TimeSpan time)

{

return Timer += time;

}

}

}

Налагоджувальний код контейнерів docker

testing\_api\_app:

container\_name: testing\_api\_app

image: busybox

volumes:

- ../app:/var/www/app/

testing\_api\_nginx:

container\_name: testing\_api\_nginx

build: builds/nginx

volumes\_from:

- testing\_api\_app

links:

- testing\_api\_php:php

volumes:

- ./logs/nginx:/var/log/nginx

ports:

- 80:80

testing\_api\_php:

container\_name: testing\_api\_php

build: builds/php-fpm

volumes\_from:

- testing\_api\_app

links:

- testing\_api\_mysql:mysql

working\_dir: /var/www/app

testing\_api\_mysql:

container\_name: testing\_api\_mysql

image: mysql:5.7

volumes:

- ./mysql:/var/lib/mysql

environment:

MYSQL\_ROOT\_PASSWORD: root

MYSQL\_DATABASE: db

MYSQL\_USER: root

MYSQL\_PASSWORD: root

Програмний код абстрактного класу AbstractApiController (реалізація функцій API)

<?php

namespace AppBundle\Controller\Api;

use FOS\RestBundle\Controller\Annotations\QueryParam;

use FOS\RestBundle\Controller\FOSRestController;

use FOS\RestBundle\Controller\Annotations\View;

use FOS\RestBundle\Request\ParamFetcherInterface;

use JMS\Serializer\SerializationContext;

use Symfony\Component\Form\FormTypeInterface;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Response;

use Symfony\Component\HttpKernel\Exception\BadRequestHttpException;

use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;

use Nelmio\ApiDocBundle\Annotation\ApiDoc;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;

/\*\*

\* Abstract Api Controller.

\*/

abstract class AbstractApiController extends FOSRestController

{

public $entity = 'BoboyanTradeBundle:Demo';

public $entityClass = '\Boboyan\TradeBundle\Entity\Demo';

public $formClass = 'Boboyan\TradeBundle\Form\DemoType';

/\*\*

\* List all objects.

\* @ApiDoc(

\* resource=true

\* )

\*

\* @QueryParam(name="offset", requirements="\d+", nullable=true, description="Offset from which to start listing objects.")

\* @QueryParam(name="limit", requirements="\d+", default="5", description="How many objects to return.")

\* @param Request $request the request object

\* @param ParamFetcherInterface $paramFetcher param fetcher service

\* @return array

\*/

public function cgetAction(Request $request/\*, ParamFetcherInterface $paramFetcher\*/)

{

$offset = $request->get('offset');

$offset = null == $offset ? 0 : $offset;

$limit = $request->get('limit');

$limit = null == $limit ? 100 : $limit;

return $objects = $this->get('doctrine')->getManager()->getRepository($this->entity)

->findBy(array()/\* $criteria \*/, null /\* $orderBy \*/, $limit, $offset);

}

/\*\*

\* Fetch an object or throw an 404 Exception.

\* @ApiDoc(

\* )

\* @View()

\* @param int $id

\* @return mixed

\* @throws NotFoundHttpException

\*/

public function getAction($id)

{

if (!($object = $this->get('doctrine')->getManager()->getRepository($this->entity)->find($id))) {

throw new NotFoundHttpException(sprintf('The resource \'%s\' was not found.', $id));

}

return $object;

}

/\*\*

\* Create a Object from the submitted data.

\* @ApiDoc()

\* @View()

\* @param Request $request the request object

\* @return FormTypeInterface|View

\* @throws BadRequestHttpException

\* @Security("has\_role('ROLE\_USER')")

\*/

public function postAction(Request $request)

{

$object = new $this->entityClass();

$form = $this->createForm($this->formClass, $object, array('method' => 'POST',));

$form->handleRequest($request);

$form->submit($request->request->all());

if ($form->isValid()) {

$em = $this->getDoctrine()->getManager();

$em->persist($object);

$em->flush();

return $object;

} else {

throw new BadRequestHttpException("Bad request");

}

}

/\*\*

\* Update object

\* @ApiDoc()

\* @View()

\* @param int $id

\* @return Response

\* @throws NotFoundHttpException

\* @throws BadRequestHttpException

\* @Security("has\_role('ROLE\_USER')")

\*/

public function putAction($id, Request $request)

{

$object = $this->get('doctrine')->getManager()->getRepository($this->entity)->find($id);

if (!$object) {

throw new NotFoundHttpException(sprintf('The resource \'%s\' was not found.', $id));

}

$form = $this->createForm($this->formClass, $object, array('method' => 'PUT'));

if ($request->getMethod() == 'PUT') {

$form->handleRequest($request);

$form->submit($request->request->all());

if ($form->isValid() && $form->isSubmitted()) {

$em = $this->getDoctrine()->getManager();

$object = $form->getData();

$em->persist($object);

$em->flush();

return $object;

} else {

throw new BadRequestHttpException("Bad request");

}

}

}

/\*\*

\* Delete Object

\* @ApiDoc()

\* @param int $id

\* @View()

\* @throws NotFoundHttpException

\* @Security("has\_role('ROLE\_USER')")

\*/

public function deleteAction($id)

{

$em = $this->getDoctrine()->getManager();

$object = $em->getRepository($this->entity)->find($id);

if (!$object) {

throw new NotFoundHttpException(sprintf('The resource \'%s\' was not found.', $id));

}

$em->remove($object);

$em->flush();

}

}