Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу

Виконав: студент 2 курсу, групи 2АКІТ-21м спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
Евеліна ГОРБАЧОВА
Ім'я ПРІЗВИЩЕ

Керівник: к.т.н, доцент, доцент кафедри КСУ
степінь, звання, посада
Олег КОВАЛЮК
Ім'я ПРІЗВИЩЕ

«12» 12 2022 р.

Опонент: к.т.н, доцент, професор кафедри АІІТ
степінь, звання, посада
Володимир ПАПІНОВ
Ім'я ПРІЗВИЩЕ

«17» 12 2022 р.

Допущено до захисту
Зав. кафедри КСУ
В'ячеслав КОВТУН
«14» 12 2022р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри КСУ

В'ячеслав КОВТУН

"03" жовтня 2022 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
студенту Горбачовій Евеліні Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу
керівник роботи Ковалюк Олег Олександрович, к.т.н., доцент
затверджені наказом ВНТУ від "14" вересня 2022 року №203
2. Термін подання студентом роботи "12" грудня 2022 року
3. Вихідні дані до роботи: програмне середовище Visual studio 2022, Office 365, Blazor, Azure Portal: Azure functions, Storage account, Service Bus, сервіси: SendGrid, адаптивні картки.
4. Зміст текстової частини: Вступ; Аналіз та обгрунтування проблеми; Аналіз меж вирішення проблеми та використаних технологій; Розробка системи автоматизації навчального процесу; Економічна частина; Висновки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) 1. Схема архітектури розробки. 2. Структури класів та зв'язків. 3. UML діаграма послідовності роботи системи. 4. Результати тестування.

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
4	Небава М.І., професор кафедри ЕПВМ		

Дата видачі завдання "03" жовтня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз методів, принципів, підходів. Постановка задачі дослідження	03.09.22	14.09.22	
2	Затвердження теми МКР	14.09.22	14.09.22	
3	Обґрунтування сучасних технологій вирішення проблеми	19.09.22	01.10.22	
4	Побудова архітектури додатку	03.10.22	04.10.22	
5	Розробка системи	05.10.22	30.11.22	
6	Розробка UML-діаграм системи	01.12.22	02.12.22	
7	Тестування системи	30.11.22	09.12.22	
8	Підготовка економічної частини	12.11.22	24.11.22	
9	Підготовка пояснювальної записки	24.11.22	12.12.11	
10	Захист МКР	20.12.22	20.12.22	

Студент _____ Евеліна ГОРБАЧОВА
(підпис) (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи _____ Олег КОВАЛЮК
(підпис) (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 004.031.42

Горбачова Е.О. Розробка інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2022 р., 149 с.

На укр. мові. Бібліогр.: 58 назв; рис.: 51**;** табл. 5.

У магістерській кваліфікаційній роботі розроблено систему інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу. У оглядово-аналітичній частині роботи розглянуто особливості побудови сучасних автоматизованих навчальних систем, а також визначена проблема і задача оцінювання знань студентів, обґрунтована доцільність розробки інтерактивних форм в поштових електронних повідомленнях. У теоретично-методичній частині визначено технології та сервіси для вирішення поставленої проблеми, обґрунтовано їх використання. Також обґрунтовано інструментальні засоби реалізації системи. У практичній частині розроблено схему архітектури системи, її класи і діаграму послідовності, вказана мапа використовуваних ресурсів. Виконано розробку програмного забезпечення та наведено результати його тестування. У економічній частині проаналізований технічний рівень і розрахована собівартість реалізації розробки.

Ілюстративна частина складається з 10 плакатів із схемами та результатами роботи.

Ключові слова: автоматизована система, навчання, інтерактивні форми, адаптивні картки, повідомлення, хмарне середовище, сервіси

ANNOTATION

UDC 004.031.42

Horbachova E.O. Development of interactive forms in postal electronic messages for automating the educational process.

Master's thesis on specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2022, 149 p.

In Ukrainian speech. Bibliography: 58 titles; Fig.: 51; tables 5.

In the master's qualification work, a system of interactive forms in postal electronic messages was developed to automate the educational process. In the review and analytical part of the work, the features of the construction of modern automated educational systems are considered, as well as the problem and task of assessing students' knowledge is defined, the reasonable feasibility of developing interactive forms in postal electronic messages is substantiated. In the theoretical and methodological part, technologies and services for solving the problem are defined, their use is justified. Instrumental means of implementing the system are also substantiated. In the practical part, a scheme of the system architecture, its classes and sequence diagram is developed, a map of the used resources is indicated. The development of the software was carried out and the results of its testing were given. In the economic part, the technical level is analyzed and the cost of development implementation is calculated.

The illustrative part consists of 10 posters with diagrams and work results.

Keywords: automated system, training, interactive forms, adaptive cards, messages, cloud environment, service.

ВІДГУК
керівника магістерської кваліфікаційної роботи

студента (-ки) Горбачової Евеліни Олександрівни
(прізвище, ім'я, по батькові)

на тему Розробка інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу

В наш час дистанційні методи навчання стають все більш актуальними, доповнюючи традиційні підходи викладання. Однією із задач, яка супроводжує онлайн-навчання, полягає в організації ефективної комунікації між слухачами курсу і лектором, менторами, координатором. Розв'язанню такої задачі і присвячена магістерська кваліфікаційна робота.

В роботі розроблено систему для комунікації через форми в поштових електронних повідомленнях для автоматизації навчального процесу.

Для реалізації системи проведено огляд існуючих аналогів та технологій розробки. Обрано стек технологій компанії Microsoft, який передбачає використання хмарних сервісів для побудови системи. Особлива увага приділена процесу неперервної інтеграції та розгортання (CI/CD).

Ефективність запропонованих рішень підтверджена експериментально.

До недоліків роботи можна віднести складність реалізації багатокрокових форм. Вказаний недолік не стосується принципових положень роботи.

Вважаю, що магістерська робота відповідає вимогам до магістерської кваліфікаційної роботи, а її автор, Горбачова Евеліна Олександрівна, заслуговує на оцінку «А» та присвоєння кваліфікації: ступінь вищої освіти магістр, спеціальність 151 – «Автоматизація та комп'ютерно-інтегровані технології», освітня програма «Інтелектуальні комп'ютерні системи».

Керівник магістерської кваліфікаційної роботи

Доцент кафедри КСУ, к.т.н., доцент _____ Олег КОВАЛЮК
(посада, науковий ступінь, вчене звання) (підпис) (*Ім'я ПРІЗВИЩЕ*)

ВІДГУК

опонента на магістерську кваліфікаційну роботу

студента (-ки) Горбачової Евеліна Олександрівни

(прізвище, ім'я, по батькові)

на тему: Розробка інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу

З розвитком дистанційних форм навчання розробка інтерактивних форм в поштових електронних повідомленнях стає все більш актуальною для автоматизації навчального процесу. Цей функціонал є необхідним, оскільки дозволяє формалізувати спосіб комунікації викладача із студентами, що дозволяє підвищити ефективність роботи, зокрема за рахунок економії часу. Тому робота є актуальною і спрямована на те, щоб автоматизувати, полегшити, пришвидшити та підвищити рівень оцінювання знань студентів.

В роботі проведено аналіз сучасних технолоній для автоматизації навчального процесу, а саме оцінювання якості знань методом тестування. Для усунення недоліків в розглянутих аналогах запропоновано архітектуру системи з використанням інтерактивних форм у поштових електронних повідомленнях, яка відрізняється від існуючих підходом розробки, технологіями та сервісами. Розроблено діаграму послідовності, що описує роботу системи, діаграму класів, та мапу використовуваних ресурсів. Проведено обгрунтувння технічних засобів для реалізації системи.

Враховуючи переваги сучасних хмарних сервісів, їх додатків та функцій, розроблено надійну серверну частину та інтерфейс системи у Azure portal від

Blazor, що дозволяє здійснювати розробку без викортстання javascript та HTML. Вся система побудована за принципом мікросервісної архітектури.

Ефективність розробленої системи підтверджена експериментально.

До недоліків системи можна віднести певну мінімальну плату за використання послуг ліцензійного Office 365 від Microsoft. Вказаний недолік не стосується принципових положень роботи. Для навчальних закладів можуть надаватися безкоштовні ліцензії.

Робота виконана із дотриманням вимог до магістерських кваліфікаційних робіт та стандартів.

Висновок: магістерська кваліфікаційна робота ***відповідає*** ~~***(не відповідає)***~~ спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» та освітньо-професійній програмі «Інтелектуальні комп'ютерні системи», заслуговує на оцінку **A**, а її автор ***заслуговує*** ~~***(не заслуговує)***~~ присудження кваліфікації: ступінь вищої освіти магістр, спеціальність 151- «Автоматизація та комп'ютерно-інтегровані технології», освітня програма «Інтелектуальні комп'ютерні системи».

Опонент

Проф. кафедри АІІТ, к.т.н., проф. _____ Володимир Папінов

(посада, науковий ступінь, вчене звання) (підпис) (*Ім'я ПРІЗВИЩЕ*)

Печатка установи, організації опонен

ЗМІСТ

ВСТУП

Завдяки сучасним можливостям комп'ютерних пристроїв та мереж, навчання стало швидким і мобільним та доступним з будь-якого куточка.

Автоматизація навчального процесу є беззаперечно популярною темою, обумовлена розвитком інформаційних технологій та карантинами, браком часу, заторами, тощо, що змінили процес звичайного офлайн навчання, на адаптивний чи повністю онлайн. Використання цифрових, інформаційних технологій дає можливість контролювання процесу навчання, коректно оцінювати студентів, взаємодіяти з викладачами дистанційно та отримувати достатній обсяг інформації для навчання.

Актуальність теми. Показником якісного навчання є точне оцінювання знань студентів, їх результат, що представляється та використовується на практиці. Оскільки, більшість викладачів досі на практиці перевіряють дані студентами відповіді у ручний та довільний спосіб, що може давати суб’єктивну оцінку (коли однакова відповідь від різних студентів може сприйматись по різному). Також вміння ведення широкої статистики рівней знань, активностей студентських груп, дає змогу загально оцінити рівень навчальної системи, рівень знань та ефективності навчання.

Мета і завдання роботи включають підвищення ефективності навчального процесу: швидка доставка інформації до кінцевих користувачів; інтеграція відправлення електронних листів вибраним групам чи користувачам виведення статистики активності студентів; використовуючи хмарні середовища. А також економія часу та ресурсів користувачів в отриманні інформації; функціональність і простота розробки у використанні.

Об’єктом дослідження є процес інтеграції відправлення інтерактивних форм в електронних повідомленнях вибраним групам чи користувачам.

Предметом дослідження є методи автоматизації навчального процесу і їх ефективне використання.

Новизна одержаних результатів – підхід розробки інтерактивних форм для електронних повідомлень, використовуючи хмарні середовища для автоматизації процесів надсилання повідомлень. Дане рішення є цілком безкоштовним, простим у застосуванні. Електронні форми можуть бути налаштовані та побудовані у будь якому форматі, який підлягатиме правилам побудови структури інтерактивної форми.

Практична цінність полягає у автоматизації навчального процесу: оцінювання знань студентів та збереження людського ресурсу викладачів , як час.

Апробація результатів магістерської кваліфікаційної роботи. Матеріали заслуховувалися на конференції ВНТУ: «Молодь в науці: дослідження, проблеми, перспективи (МН-2022)» [1].

Публікації результатів магістерської кваліфікаційної роботи. Матеріали опубліковувались тезами до конференції ВНТУ: «Молодь в науці: дослідження, проблеми, перспективи (МН-2022)» [1].

1 АНАЛІЗ ТА ОБҐРУНТУВАННЯ ПРОБЛЕМИ

Причини того, чому з’явились перші системи управління навчанням (СУН) стали саме університети, адже – велике число студентів сприяло величезному навантаженню на навчальні відділи, але одночасно вони ж забезпечували і ресурс для розробки програмного забезпечення [2].

Автоматизованою навчальною системою (АНС) розуміють узгоджену сукупність навчальних матеріалів, засобів їх підготовки та розробки, зберігання інформації, обмін та передача доступу до них. Має на меті автоматизований процес навчання, заснований на використанні сучасних інформаційних технологій [3].

Основні функції АНС [3]:

- доступ до освітніх ресурсів (ОР), включаючи кошти віртуальних і віддалених навчально-дослідних лабораторій;
- самотестування і контроль знань учнів;
- пошук інформації;
- створення ОР;
- управління навчальним процесом;
- конференц - зв’язок (чати, потокове відео).

Автоматизована навчальна система [3]:

- контролює права доступу користувачів;
- здійснює пошук необхідних матеріалів, витягує їх з баззи учбових матеріалів (БУМ) і надає користувачеві;
- забезпечує доступ до індивідуальної робочого зошита, що містить графік навчальних занять, результати виконання навчальних завдань та інші замітки користувача, пов’язані з вивченням курсу;
- реалізує телекомунікаційні зв’язку "викладач-якого навчають", "студент-лабораторія" і "навчений-якого навчають" і зв’язок

користувачів з Internet для пошуку інформації, організації конференцій та спільної роботи над проектами;

- допомагає адміністратору підтримувати систему в актуальному стані, вести облік користувачів та ін.

Навчальна система є складною системою, під час створення якої вже на початковому етапі необхідно забезпечити можливість її розвитку, модифікації і вдосконалення [4].

1.2 Проблема оцінювання знань студентів (зріз знань, тестування)

Застосування інтернет технологій (ІТ) докорінно змінило навчальний процес. Автоматизація освітньої галузі є одним з найважливіших завдань сьогодні.

Найскладнішим етапом є оцінювання знань студентів, це необхідна умова вищих навчальних закладів. Зрізи, тестування чи рубежі знань слугують для актуалізації знань студентів, встановлення рівнів успішності академічних груп та окремих студентів, аналізу різних форм і методів навчання, підсумкового оцінювання. “Тестування є одночасно і навчальною вправою, і засобом контролю. Використання тестових завдань як засобу навчання є ефективним і виправданим з точки зору дидактики та психології. Чітка регламентованість процедури тестування та наявність еталону дозволяють оптимізувати навчальний процес, а спрямованість на активну розумову діяльність та об’єктивність результатів створюють у студентів позитивне ставлення до (предмету)” [5].

Тест — це завдання стандартної форми, що має за мету визначити рівень засвоєння знань, розумового розвитку, спеціальних можливостей та інших якостей особистості людини [6].

Виділяють такі види тестів [7]:

- Тест, як проміжний контроль успішності;
- Тест підсумковий контролю успішності.

За формальними ознаками (за структурою та способом оформлення тесту):

- з вибором однієї чи кількох правильних відповідей;
- на встановлення відповідності (утворення логічних пар);
- відкриті завдання з короткою чи розгорнутою відповіддю.

Педагогічні тести виконують наступні функції [8]:

- Діагностична. Функція зворотного зв'язку, передбачає виявлення та оцінювання якостей особистості.
- Навчальна. Реалізується шляхом активізації навчальної діяльності щодо засвоєння навчального матеріалу.
- Виховна. Виявляється у формуванні позитивних якостей особистості, як: інтерес до знань, уміння систематично працювати, навички самоконтролю та самооцінки, активність, почуття самоповаги.
- Розвивальна. Виявляється у розвитку пам'яті та мислення, формуванні
- умінь і навичок використання знань на практиці.
- Прогностична. Дає змогу передбачити потенційні можливості тих, кого
- навчають, в засвоєнні нового матеріалу.
- Мотиваційна. Здійснює мотивацію до навчального процесу.
- Контрольна. Слугує для попереднього, поточного, рубіжного та підсумкового контролю якості знань.

Закриті тестові завдання (ТЗ) [9]:

За вибором:

- дихотомічний;
- множинний;

- на встановлення відповідності;
- на встановлення правильної послідовності;

Відкриті ТЗ:

- на коротку відповідь;
- на доповнення;
- на розгорнуту відповідь:
- структуровані;
- неструктуровані.

Якість вищої освіти великою мірою залежіть від стану контролю навчальних досягнень студентів [5].

Якість освіти сучасного типу можуть забезпечити лише сучасні методи контролю навчальних досягнень, зокрема, комп'ютерне тестування. Найпопулярнішим видом такого контролю є тестування, засноване на діалозі обчислювальної системи з користувачем. "Комп'ютерне тестування — ідея комп'ютерного тестування прямо виникає від ідеї програмованого контролю знань [10].

Комп'юторне оцінювання знань покриває такі проблеми ручної перевірки [11]:

- скорочення часу обробки великого обсягу різноманітного навчального матеріалу численної групи опитуваних;
- регулювання попередньо визначеного рівня вимог, допускаючи автоматичну зміну ступеня складності запитань;
- самоконтроль на попередньому етапі з метою самооцінювання результатів підготовки перед офіційним тестуванням;
- отримання об'єктивної оцінки без врахованого людського фактору;

- організація зворотного зв’язку між студентом і викладачем з використанням мережі інтернет;
- формування узагальнених статистичних оцінок результатів контролю, та процесу навчання.

Ряд актуальних недоліків комп’ютерного тестування при формуванні тестів з відображенням декількох альтернативних варіантів відповідей відмічених в роботі [12], зокрема:

- наявність тільки однієї правильної відповіді;
- вибір правильної відповіді навмання чи навздогад;
- відсутність доступності до самостійного формування відповіді;
- оцінювання розширених запитань дає можливість обробити лише кінцевий результат;
- необхідність формулювань тестових завдань за допомогою висококваліфікованих фахівців і експертах.

З початком нової віхи в дистанційному навчанні пов’язані можливості використовувати адаптовані технології та методики тестування з використанням всесвітньої мережі. Це допомагає піддтримувати інтерактивний діалог зі студентами, здійснювати контроль і підтримку в режимі реального часу, удосконалювати стратегію навчання і тестування на основі визначеного рівня індивідуальних знань, навичок і здібностей того, кого навчають [13].

Саме тому було запропоновано вирішення задачі організації взаємодії з користувачами, яке економить час та ресурси викладачів, є функціональним і простим у використанні. Опираючись на головну проблему і задачу дослідження, було запропоновано розробку комп'ютерної системи для автоматизації навчального процесу: інтеграція інтерактивних форм в електронних повідомленнях, сайт з підтримкою технології реального часу, обґрунтовано вибір інструментів розробки,

розпочато організацію середовища проекту та імплементацію системи з використанням Actionable Messages [1]. Використання даних технологій забезпечить економічність, збереження такого ресурсу, як час, також відкине потребу додаткових фінансових витрат на папір, канцелярію, тощо.

1.3 Аналіз аналогів розробки

Одним із стандартних методів оцінювання якості знань студента є тестування. Кожен спосіб вимірювання якості знань, що використовуєтьмя сьогодні, має ряд недоліків та переваг, які саме і впливають на оцінку знань. Застосування усних і письмових контрольних робіт, рефератів, тощо стльніше та частіше схиляються до суб'єктивної оцінки знань студента викладачем. Тестування ж дає змогу оцінити студента об'єктивно.

Для аналізу пропонується 3 аналога автоматизованих систем тестування.

1. Тестові оболонки у MACROMEDIA AUTHORWARE, розроблені для тестування студентів. [14].

Основними стандартними засобами, які призначені для побудови системи тестування є набір визначених об'єктів, які створенні розробниками Macromedia Authorware. Ці об'єкти об'єднані у спеціальний розділ бібліотеки об'єктів, який називається Assessmeny (Оцінка). Доступ до цього розділу здійснюється за допомогою вікна Knowledge Objects.

Більша частина об'єктів цієї категорії призначена для реалізації основних типів тестів: Drag-Drop Question - забезпечує реалізацію тесту маніпулювання об'єктами, які студент повинен перемістити у відповідності з деяким правилом;

Hot Object Question - забезпечує реалізацію тесту, в якому студент повинен вибрати один або декілька об'єктів у відповідності з деякою ознакою;

Hot Spot Question - забезпечує реалізацію тесту, де на одному графічному зображенні студент повинен вибрати відповідну область;

Multiple Choice Question - забезпечує реалізацію тесту багато з багатьох, де студент повинен вибрати усі вірні відповіді з тих, що запропоновані;

Short Answer Question - реалізує відкритий тест, типу заповнення бланка з контролем по ключовим словам;

Single Choice Question - забезпечує реалізацію тесту «один з багатьох», де студент повинен вибрати один (вірний) з декількох запропонованих відповідей;

True-False Question - забезпечує реалізацію бінарного вибору;

Scoring - активізує та налагоджує підсистему визначення підсумкових оцінок по наслідкам тестування та формування звіту.

Дана програма є консольною, запускається лише на комп’ютері:

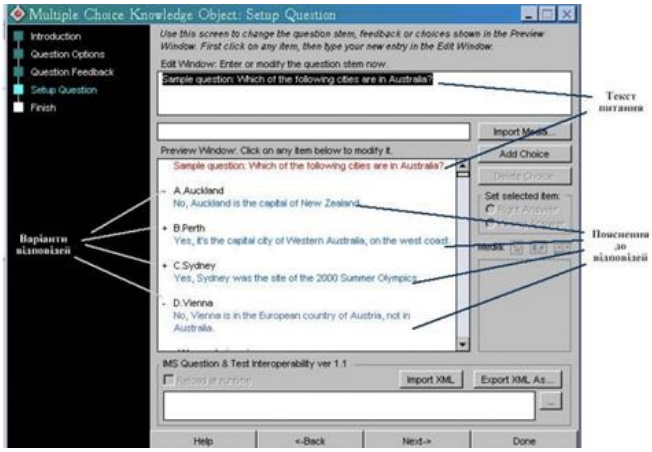


Рисунок 1.1. Вікно майстра на третьому кроці Question Question

Також варто віднести до недоліків, ще мову інтерфейсу, тобто вікна майстрів всіх заздалегідь визначених об’єктів некоректно відображають текст на українській та російській мовах, оскільки при створенні таких об’єктів розробниками були застосовані шрифти без підтримки кирилиці. Однак при відтворенні вікна з тестом, введений текст буде відображатись вірно при умові, що були модифіковані стилі тексту.

2. Система MOODLE для проведення елктронного тестування.[15].

Система управління навчанням MOODLE (акронім від Modular ObjectOriented Dynamic Learning Environment – це модульне об’єктно-орієнтоване динамічне навчальне середовище), що призначена для організованої інтеграції викладачів, адміністраторів і студентів в одну надійну та безпечну систему [16]. Завдяки розширеному функціоналу як для студентів (доступність навчальних матеріалів; наявність засобів для групової роботи; можливість перегляду результатів тестування та успішності освоєння дистанційних курсів), так і для викладачів (наявність інструментів для розробки авторських дистанційних курсів; можливість додавання різноманітних елементів курсу; швидка модифікація матеріалів; можливість використання різних типів тестів; автоматизація процесу перевірки знань, формування звітів щодо успішності проходження курсів тощо) система MOODLE набула значної популярності і успішно використовується у більш, ніж 200 країнах світу.

Серед функціональних можливостей системи для електронного опитування слід відмітити інструмент Тест, що дозволяє створити різноманітні види тестів (рис.1.2).

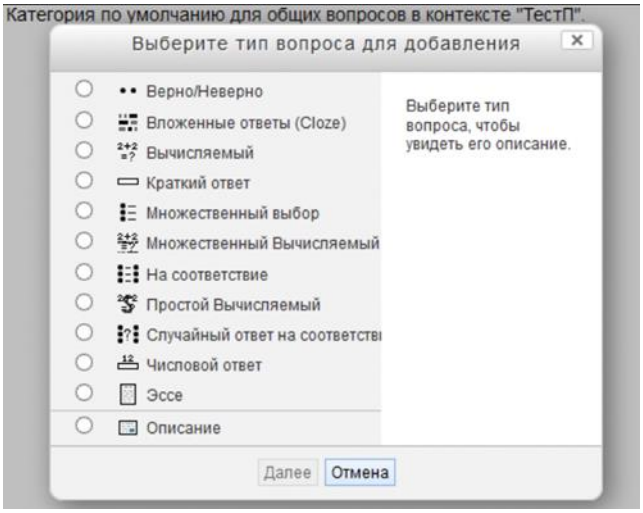


Рисунок 1.2. Інструменти системи MOODLE для створення тестів

Таким чином, використання системи управління навчанням MOODLE для проведення опитування знань студентів дозволяє отримати ряд переваг, серед яких:

- забезпечення прозорості процесу опитування;
- можливість дистанційного проходження курсів та оцінювання засвоєних знань;
- полегшення праці викладача за рахунок усунення необхідності перевірки контрольних робіт та тестів;
- підвищення якості навчального процесу.

Як недоліки можна підкреслити, доступ до тестування лише з комп'ютера, на якому встановлена програма MOODLE.

3. MyTest [17]

MyTest [17] – це система об'єднаних програм (програма тестування учнів, редактор тестів і журнал результатів), що відповідає створенню і проведенню комп'ютерного тестування, збору і аналізу результатів, оцінювання за вказаною шкалою з тесту. Програма легка, зручна у використанні і безкоштовна. MyTest працює з різними типами завдань: одиничний вибір, вибір з множиною відповідей, встановлення порядку проходження, встановлення відповідності, вказівка істинності або не істинності тверджень, ручне введення числа у рядок, ручний набір тексту, виділення зображення. Кожен відкритий у системі тест є незалежною програмою. Створені файли тестів надалі можуть бути запущені на будь-якому комп'ютері без встановлення додаткових програм чи додатків. Програма дає можливість суміщати питання трьох типів: одиночний вибір правильної відповіді із списку запропонованих, множинний вибір або набір відповіді з клавіатури в одному тесті. Кожне питання або варіант відповіді може містити текст, малюнки, формули, діаграми і ін.

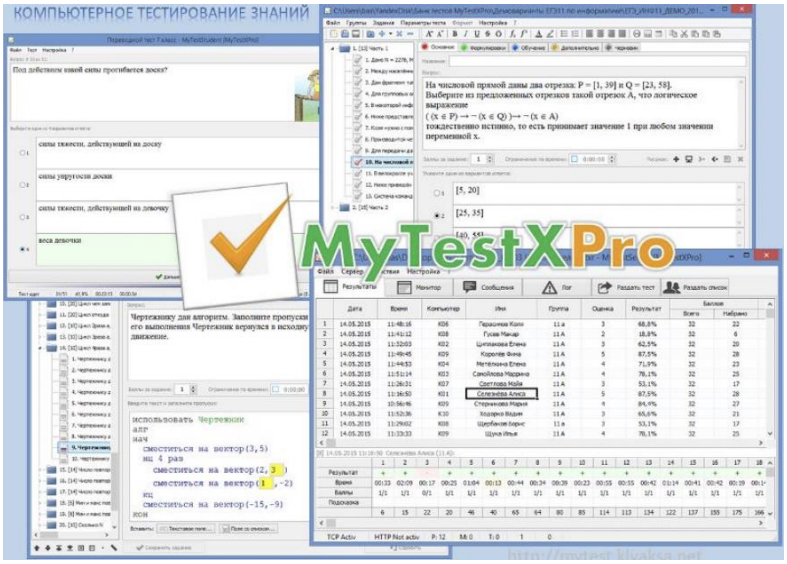


Рисунок 1.3 Скрін програми MyTest

4. SurveyMonkey [18].

Також одним із найпопулярніших аналогів є SurveyMonkey - платформа для створення та проведення онлайн-опитування, а також масової розсилки анкет та виявлення тенденцій. Сервіс дозволяє швидко створювати опитування, налаштовувати їх зовнішній вигляд, проводити A/B-тестування, інтегрувати опитування на сайти та в соціальні мережі, складати дуже детальні та наочні звіти, захищати дані [19].

Функції SurveyMonkey:

- Опитування.
- Вибір шаблонів опитувань.
- Підтримка всіх мов.
- Логіка сторінок та питань.
- Теми користувача та звіти.
- Сортування варіантів відповідей.
- Індикатор пройдених питань.

- Автонумерація сторінок та питань.
- Версія PDF для друку.
- Надсилання опитування за веб-посиланням, поштою, у Twitter або Facebook.
- Настроювана URL-адреса.
- Розсилання опитувань диспетчером.
- Підвищена безпека.
- Підсумки у режимі реального часу.
- Аналітика тексту.
- Спільний доступ до відповідей.

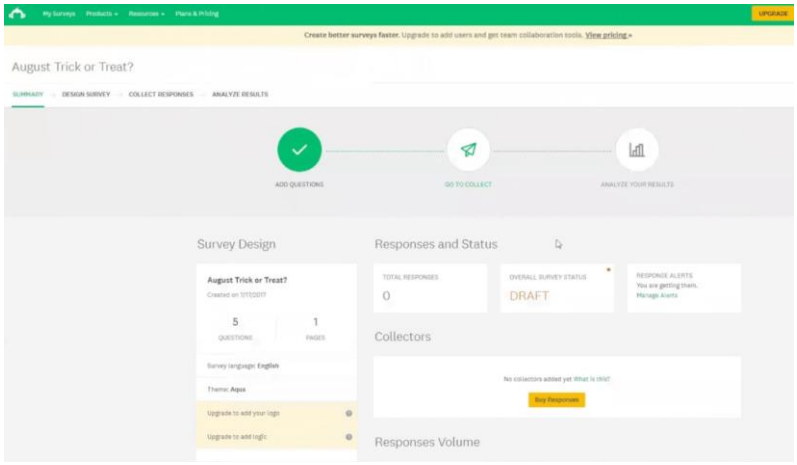


Рисунок 1.4 – Інтерфейс системи

У SurveyMonkey — широкий банк вбудованих питань різних типів, що налаштовуються: задоволеність клієнтів, зворотний зв'язок, демографічні дані, маркетингові дослідження, HR-анкетування, фідбек про заходи[20].

Більшість із вказаних функцій відсутня у безкоштовній версіїї, яка є дуже обмеженою - у формі доступно лише 10 питань та 100 відповідей до них.

Розглянуті аналоги покривають лише частину актуальних проблем користувачів.

Запропоновий у ході дослідення варіант розробки інтерактивних форм у електрогих поштових повідомленнях включає в себе різноманнія варіацій представлення тестових завдань, розсилання методичного матеріалу. Також викладач сам контролює кількість спроб відповідей та час на виконання завдання, головне - мобільність, для його виконання не потрібно встановлювати ніяких додатків чи бути за комп'ютером, достатньо відкрити надісланого електронного листа. Дана розробка не буде обтяжувати файлами чи громізткими старими програмами. Додатково, буде розроблемо "Веб потрад викладача", для перегляду статики активності студентів.

1.4 Постановка задачі дослідження

Оскільки завданням розробки є імплементація інтерактивних форм в поштових електронних повідомленнях, можна виділити такі задачі для виконання:

- проаналізувати важливість та необхідність оцінювання знань студентів;
- проаналізувати існуючі засоби та методи для автоматизації навчального процесу;
- порівняти можливості хмарних середовищ та ресурсів для реалізації інтерактивних форм;
- розробити архітектури додатку;
- розробити веб застосунок для створення та редагування форм опитування
- розробити функцію відправки електроних повідомлень;
- інтегрувати систему авторизації для входу в веб застосунок;
- розробити веб портал викладача для перегляду статистичних даних;
- розробити реалізацію підрахунку та відображення статистичних результатів;

- розробити автоматичне розгортування системи.

1.5 Висновки

В даному розділі проаналізовано сучасний стан освітньої системи, розвиток автоматизації даної галузі. Визначено перелік існуючих проблем оцінювання якості знань, що показав доцільність створення програмного продукту. На базі проведеного аналізу предметної області сформульовано задачі дослідження, які потребують вирішення в ході виконання роботи.

2 АНАЛІЗ МЕЖ ВИРІШЕННЯ ПРОБЛЕМИ ТА ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Визначивши недоліки використання звичайного тестування, анкетування і проведення контролю чи зрізу знань студентів виникає наукова задача, яка полягає в підборі та розробці методів засобів систем автоматизованого навчання, які могли

б забезпечувати розв’язок задачі якісного оцінювання знань та управління навчальним процесом [21].

Оскільки, вищі навчальні заклади, такі як ВНТУ, витрачають велику кількісь ресурсів, такі як час на підготовку, збір та опрацювання на анкетування, фінансування канцелярії, ведення статистики груп, кафедер та відділів, важливо визначити основні функції автоматизованого оцінювання.

Автоматизована оцінка знань студентів в системі управління навчанням включає в себе такі проекції:

1. Результати тестування знань студентів.
2. Рейтинг активності роботи студентів в електронному середовищі.
3. Гейміфіковані модулі мотивації студентів (дошка пошани, призи, додаткові бали тощо) [22].

Сьогодні, у часи активного використання ІТ-технолоній, світових змін особливо важливо забезпечити якісне оцінювання знань студентів, так як, існує проблема мобільності. Мобільність навчання - доступ до навчання з різних гаджетів, у зручний спосіб, економлячи час та ресурси.

Слід звернути увагу на таку проблему, як списування чи перепроходження контрольних робіт декілька разів. Крім того, не всі програми підтримують змогу відповіді відкритого типу, декільуа варіантних чи містити графічний/відео/аудіо завдання чи методичні вказівки до виконання роботи.

Визначивши обсяг проблеми, конкретні цілі та функції, що очікуються до впровадження, пропонується вирішення проблеми з використанням наступних методів та новітніх технологій: Microsoft Actionable Messages від Azure забезпечує створення та контроль інтерактивних елктроних повідомлень. Ці повіломдення можна налаштувати під різні потреби тестування, звичайного антекутвання, а також навчання. Проходити контрольну роботу у один клік з телефону. Усі відповіді

збираються та обробляються, виводяться у графіки успішності та активності студентів, що доступні для викладачів.

Усі ці функції є екогомічними, вони працють за технологією Serverless (оплачуючи лише той час, коли працює сервіс) MS office 365, Azure Devops, Azure продукти та Blazor.

2.1 Адаптивні картки (Microsoft Adaptive cards)

Microsoft Adaptive cards (адаптивні картки) — це відкритий формат обміну даними, який забезпечує уніфіковану та узгоджену передачу вмісту інтерфейсу користувача між розробниками [23].

Автори карток описують свій вміст як простий об'єкт JSON. Для цього вмісту можна виконувати власне відображення всередині провідного додатка з автоматичною адаптацією до його інтерфейсу.

Наприклад, бот Contoso може створити адаптивну картку за допомогою Bot Framework, а після доставки цієї картки до Skype вона буде оформлена відповідно до інтерфейсу Skype. Якщо корисні дані надіслати до Microsoft Teams, вони будуть адаптовані до інтерфейсу Microsoft Teams. Коли більше провідних програм почнуть підтримувати адаптивні картки, ті ж корисні дані автоматично підсвічуватимуться в таких програмах. Але дані, як і раніше, будуть повністю адаптовані до інтерфейсу програми [23].

Користувачі в цей час працюють зі звичним інтерфейсом. Провідні програми управляють взаємодією з користувачем.

Цілі, що досягаються за допомогою адаптивних карток [23]:

- Портативність. Картки підходять для будь-якої програми, пристрою та інфраструктури інтерфейсу користувача.

- Відкритий код. Бібліотеки та схеми надаються з відкритим кодом та у загальнодоступній версії.
- Низька вартість. Картки прості у розробці та використанні.
- Виразність. Картки відображають найрізноманітніший вміст, який створюють розробники.
- Повна декларативність. Код не потрібний і не допускається.
- Автоматична стилізація. Картки адаптуються до інтерфейсу провідного додатка та стилю бренду.

Переваги адаптивних карток [23]:

- Узгоджений інтерфейс користувача. Ви гарантуєте узгодженість оформлення і особливостей використання, оскільки картки малюються в стилі інтерфейсу, що належить вам.
- Відповідність внутрішнім показникам продуктивності. Ви забезпечуєте відповідність своїм показникам продуктивності, так як картки орієнтовані безпосередньо на вашу інфраструктуру інтерфейсу користувача.
- Безпека. Вміст доставляється у вигляді захищених корисних даних, щоб ви не відкривали свою інфраструктуру інтерфейсу для необроблених даних розмітки та скриптів.
- Простота реалізації. Ви отримуєте готові бібліотеки для простої інтеграції на будь-якій підтримуваній платформі.
- Безкоштовна документація. Ви заощаджуєте час, тому що вам не потрібно розробляти, реалізувати та документувати власну схему.
- Загальні засоби. Ви заощаджуєте час, тому що вам не потрібно створювати спеціальний інструментарій.

2.2 Інтерактивні повідомлення (Microsoft Actionable Messages)

Microsoft Actionable Messages (Інтерактивні повідомлення) є частиною Microsoft Adaptive cards, вони дозволяють розширити функціональність повідомлень електронної пошти в Outlook, щоб дати одержувачам можливість швидко виконувати дії за допомогою адаптивної картки, не виходячи з Outlook [24].

Actionable Messages дають можливість організаціям створювати електронні листи, які дозволяють виконувати певні завдання, не залишаючи папки «Вхідні». Зазвичай вони використовуються для внутрішнього зв'язку та можуть дозволити користувачам Outlook [25].

Actionable Messages спрямовані на підвищення ефективності електронної пошти за рахунок зменшення кількості взаємодій, необхідних для виконання певного завдання.

Завдяки Actionable Messages окрема особа або група отримує електронний лист із усіма інструментами (кнопками, текстовими полями, спадними меню тощо), які їм потрібні для виконання завдання. Ця інформація відображається в «Адаптивній картці», налаштованої відповідно до конкретних вимог. Дані, зібрані з цих відповідних повідомлень, потім передаються назад автору електронного листа та стають доступними безпосередньо в його поштовій скриньці [25].

Приклади Actionable Messages форм у роботі та електронних листах користувачів[19]:

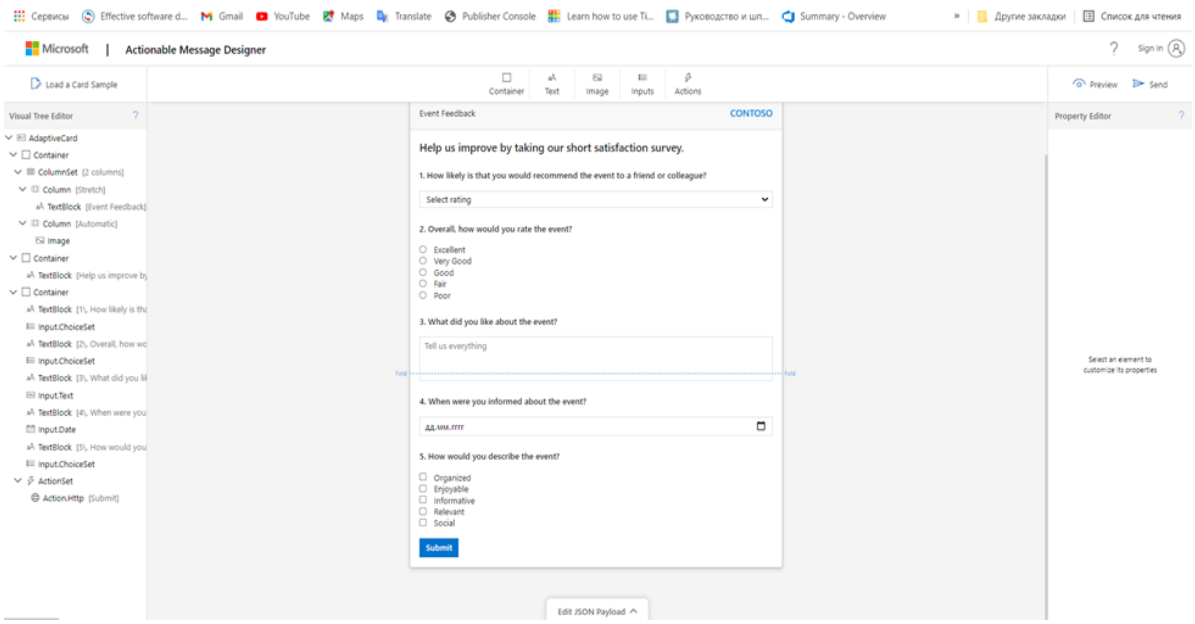


Рисунок 2.1 – Скрін створення форми для тестування

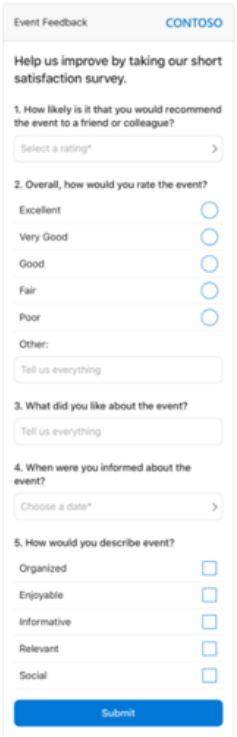


Рисунок 2.2 – Скрін форми тестування у електронному повідомленні

2.3 Хмарне середовище Azure

У ході аналізу висвітленої проблеми та пошуку рішень було обрано продукти від Microsoft, оскільки дана компанія є однією із маштабніших та найпопулярніших по розробці програмного забезпечення. Надаючи широкий вибір продуктів та засобів для розробки із необмеженим розширенням функцій та за мінімальну ціну, підтримує активний розвиток програмованих продуктів та нових течій.

Так як, представлені Microsoft Adaptive cards та Microsoft Actionable Messages найкраще будуть поєднуватись із тими самими продуктами Microsoft, було проаналізовано великий послуг. Найбільш актуальною темою сьогодні є швидкість, зручність, економічність та ергономічність розробок, де витрачається мінімум ресурсів, щоб досягти високого результату було обрано роботу із хмарним середовищем, від Microsoft пропонується Azure.

Azure — це платформа хмарних обчислень із набором послуг, що постійно розширюється,задля створення рішень для досягнення різних бізнес-цілей. Сервіси Azure варіюються від простих веб-сервісів, до роботи повністю віртуалізованих комп'ютерів, якы надають можливість запускати власні програмні рішення. Azure надає велику кількість хмарних послуг, як-от віддалене сховище, розміщення баз даних і централізоване керування обліковими записами. Azure також пропонує нові можливості, такі як AI та Інтернет речей (IoT) [26].

Хмарні обчислення або "хмара" — це надання обчислювальних послуг через Інтернет із використанням моделі ціноутворення «оплата за використання». Ці послуги включають сервери, сховище, бази даних, мережі, програмне забезпечення, аналітику та розвідку. Хмарні обчислення пропонують швидші інновації, гнучкі ресурси та економію за рахунок масштабу. Зазвичай оплачується лише хмарні служби, які використовуються, це допомагає [26]:

- Знизити експлуатаційні витрати.
- Керувати інфраструктурою ефективніше.

- Розширення інфрастуктури залежно від потреб бізнесу.
- Представляється сховище даних.
- Швидке вирішення найскладніших бізнес-завдань.

Хмарне середовище Azure допомагає рухатися швидше та впроваджувати інноваційні способи, що колись були майже неможливими.

У нашому цифровому світі, що постійно змінюється, з’являються дві тенденції:

- Команди надають користувачам нові функції з рекордною швидкістю.
- Користувачі очікують все більш насиченого та захоплюючого досвіду роботи зі своїми пристроями та програмним забезпеченням.

Колись випуск програмного забезпечення планувався на місяці чи навіть роки. Сьогодні команди випускають функції невеликими партіями, які часто розраховані на кілька днів або тижнів. Деякі команди навіть безперервно постачають оновлення програмного забезпечення – іноді з кількома випусками протягом одного дня.

Azure надає понад 100 служб, які дають змогу виконувати будь-які дії: від запуску наявних програм на віртуальних машинах до вивчення нових парадигм програмного забезпечення, таких як інтелектуальні боти та змішана реальність [26].

Портал Azure — це уніфікована веб-консоль, яка є альтернативою інструментам командного рядка. На порталі Azure ви можете керувати своєю підпискою на Azure за допомогою графічного інтерфейсу користувача.

У користувацькому доступі є різні сховища даних і мережевих компонентів, також розпізнавання мовлення та інші когнітивні служби, які допомагають виділити програму з по-між тисячі інших.

А сервіси аналітики, які надають телеметричні дані з вашого програмного забезпечення та пристроїв, допомагають швидко виявляти різні проблеми та вирішувати їх [26].

2.3 Azure веб сервіси - Azure Web Service & Azure Functions

Azure допомагає у вирішенні складних задач, він пропонує велику кількість сервісів для розробки та покращення продукту. Опираючись на задачу дослідження, одними із основних моментів є розробка веб застосунку та функція відправки електронних повідомлень. Тому слід розглянути використання Azure Web Service та Azure Functions, що саме відповідає поставленій задачі.

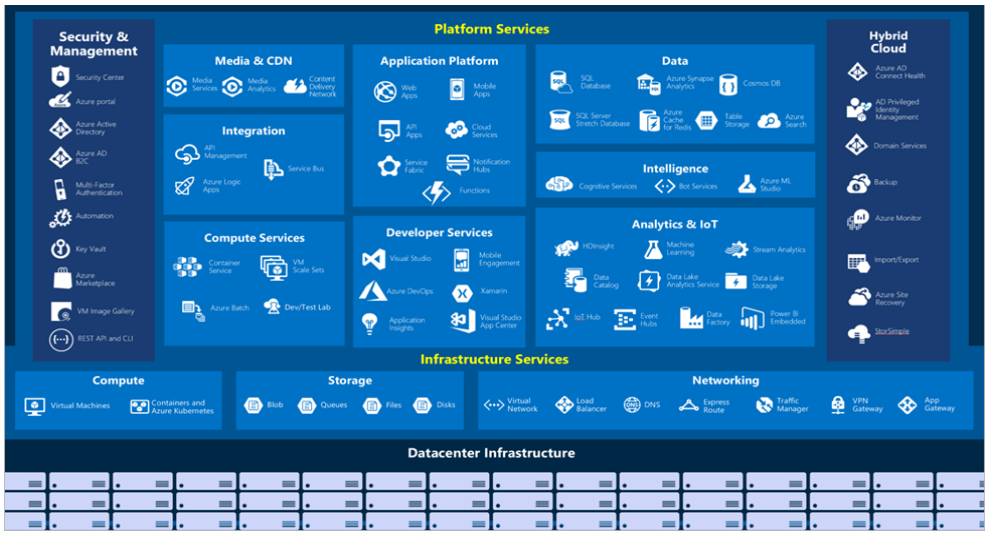


Рисунок 2.3 - Зображення найвикористовуваніших сервісів Azure

У сучасному діловому світі критично важливо мати чудовий веб-досвід. Azure включає першокласну підтримку для створення та розміщення веб-програм і веб-служб на основі HTTP [26].

Azure Web Service або служба додатків Azure – це служба на базі HTTP для розміщення веб-додатків, інтерфейсів REST API та серверної частини мобільних рішень. Ви можете виконувати розробку звичною мовою: .NET, .NET Core, Java,

Ruby, Node.js, PHP або Python. Програми легко працюють і масштабуються в середовищах на основі Windows і Linux [27].

У службі програм реалізовані не тільки можливості Microsoft Azure для програми, включаючи функції безпеки, балансування навантаження, автоматичного масштабування та автоматизованого керування. DevOps, як безперервне розгортання з Azure DevOps, GitHub, Docker Hub та інших джерел, управління пакетами, а також можливість використання проміжних середовищ, особистого домену та сертифікатів TLS/SSL [27].

Служба додатків Azure – це повністю кероване рішення на основі моделі "платформа як послуга" (PaaS) для розробників. Нижче наведено деякі ключові функції служби додатків [27]:

- Підтримка кількох мов і платформ. Служба програм повністю підтримує ASP.NET, ASP.NET Core, Java, Ruby, Node.js, PHP і Python. Можливий запуск PowerShell та інші скрипти або використані файли в якості фонових служб.
- Керована робоча середа — Служба додатків автоматично встановлює інтерфейси і забезпечує обслуговування ОС і мовних платформ.
- Контейнерізація та Docker — Запуск програм з кількома контейнерами за допомогою Docker Compose. Доступна робота з Docker безпосередньо в Службі програм.
- Оптимізація DevOps — налаштування безперервної інтеграції за допомогою безперервного розгортування з Azure DevOps, GitHub, BitBucket, Docker Hub або Reestra контейнерів Azure. Управління додатками в службових програмах за допомогою оболонки Azure PowerShell або кросплатформенного інтерфейсу командного рядка (CLI).

- Високодоступне глобальне масштабування. Зберігайте програми в будь-якому місці глобальної інфраструктури центру обробки даних. При цьому рівень обслуговування гарантує високу доступність.
- Підключення до платформ SaaS і локальним даним. Доступно понад 50 сполучників для корпоративних систем (наприклад, SAP), служб SaaS (наприклад, Salesforce) і популярних інтернет-служб (наприклад, Facebook).
- Безпека і відповідність вимогам. Служба додатків сумісна зі стандартами ISO, SOC і PCI. Доступна аутентифікація користувачів за допомогою Azure Active Directory, Google, Facebook, Twitter або в обліковому записі Майкрософт. Налаштування обмеження IP-адрес і служб ідентифікацій захищає дані.
- Шаблони розробки додатків. Доступне використання додатків - шаблонів із великого списку в Azure Marketplace, наприклад WordPress, Joomla та Drupal.
- Інтеграція з Visual Studio та Visual Studio Code. Виділені інструменти в Visual Studio та Visual Studio Code сприяють створенню, розгортанню та відлагоджуванню програм.
- Функції API і мобільних додатків. Служба програм забезпечує повну підтримку CORS для роботи з RESTful API. Також вона спрощує використання мобільних додатків, забезпечує аутентифікацію, автономну синхронізацію даних, відправку push-увідомлень і багато іншого.
- Служба є незалежною від коду сервера. Фрагменти коду або скрипта за вимогою без необхідності явно підготовлювати та адмініструвати

інфраструктуру. Платіть тільки за час виконання коду (см. статтю Документація по функціям Azure).

- Сервіс Azure SignalR. Дозволяє додавати веб-функції в реальному часі.

Крім Служби додатків, Azure пропонує й інші служби, які можна використовувати для розміщення веб-сайтів і веб-додатків.

Azure Functions або Azure функції — це хмарна служба, за доступом на вимогу, що надає всю інфраструктуру та ресурси, постійно оновлюються, необхідні для роботи розроблюваних програм. Розробник зосереджуєтеся на коді, який є найбільш важливий, написаний на зручній мові, а функції впораються з усім іншим. Функції забезпечують безсерверні обчислення для Azure. Їх можна використовувати для створення веб-інтерфейсів API, реагування на зміни бази даних, обробки потоків IoT, керування чергами повідомлень, тощо[28].

Azure Functions представляють безсерверне рішення, яке дозволяє писати менше коду, обслуговувати менше інфраструктури та заощаджувати кошти, тобто плата за сервыси буде лише тоды, коли вони будуть використовуватись. Також хмарна інфраструктура надає всі актуальні ресурси, необхідні для забезпечення роботи ваших програм [29].

Функції Azure дозволяють реалізувати логіку вашої системи в готові блоки коду. Ці кодові блоки називаються «функціями». Різні функції можуть запускатися в будь-який час, коли потрібно реагувати на критичні події. Зі збільшенням запитів Azure Functions задовольняє попит стільки ресурсів і екземплярів функцій, скільки необхідно, але лише за потреби. Коли кількість запитів зменшується, будь-які додаткові ресурси та екземпляри програм автоматично припиняються[29].

Аналогами Azure Web Service та Azure Functions є такі сервіси [30]:

- AWS Elastic Beanstalk: Після завантаження прогруми, Elastic Beanstalk автоматично обробляє деталі розгортання, пов’язані з наданням

ємності, балансуванням навантаження, автоматичним масштабуванням і моніторингом працездатності програми.

- Google App Engine: Google має репутацію високонадійної та високопродуктивної інфраструктури. За допомогою App Engine можна виконувати запуск масштабованих систем, керованих продуктивністю. Програми App Engine легко створювати, обслуговувати та легко масштабувати в міру зростання потреб у трафіку та сховищі даних.
- Kubernetes: Kubernetes — це система оркестровки з відкритим кодом для контейнерів Docker. Він обробляє планування на вузлах обчислювального кластера та активно керує робочими навантаженнями, щоб переконатися, що їхній стан відповідає заявленим користувачами намірам.

Розглянувши аналоги та безпосередньо Azure Web Service та Azure Functions можна зробити висновок, що продукти та програми Microsoft, краще інтегруються між собою, а перелік проаналізованих сервісів та функцій, сприяють покращеню та полегшеню розробки.

2.4 Обґрунтування використання мови програмування C# та .Net 6

Така мова програмування як C#, сьогодні, є однією з найпопулярніших мов, потужних мов програмування, що швидко розвиваються та є затребуваними в IT-галузі. Зараз на C# розробляються різні програми, що показують високу надійність коду та якість, зазвичай програми розділяють: на невеликі десктопні програми та великі веб-портали та веб-сервісів, що обслуговують щодня мільйони користувачів [32]. Мова C# неймовірно гнучка, завдяки чому з її допомогою можна розробляти

різні продукти для Windows, Android і iOS, працювати з файловими системами, забезпечувати безпеку та ін [31].

Перша версія мови вийшла разом із релізом Microsoft Visual Studio .NET у лютому 2002 року. Поточною версією мови є версія C# 11, яка вийшла 8 листопада 2022 разом із релізом .NET 7.

C# є мовою із Сі-подібним синтаксисом, що схожий у цьому відношенні до вимогливо типізованих C++ та Java [32].

C# є об'єктно-орієнтованою мовою, підтримує поліморфізм, успадкування, навантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід вирішує різноманітні задачі з побудови великих, але в той же час гнучких, масштабованих додатків, що можуть розширюватись. Мова C# активно розвивається, тому з кожною новою версією з'являється все більше цікавих та корисних функціональностей [32].

Переваги C# [33]:

- C# - це об'єктно-орієнтована, проста і в той же час потужна мова програмування, яка дозволяє розробникам створювати багатофункціональні програми.
- C# відноситься до мов компілюваного типу, тому вона має всі переваги таких мов.
- Широка сфера застосування.
- C# підтримує велику кількість інструментів і фреймворків. Через велику різноманітність синтаксичних конструкцій та можливості працювати з платформою .Net, C# дозволяє швидше, ніж будь-яку іншу мову, розробляти програмні рішення.
- Мова створена так, щоб дати розробникові максимальний контроль над усіма аспектами виконання програми [31].

- Мова активно розвивається, нові поліпшення з’являються набагато швидше, ніж в інших мовах програмування [31].
- C# відрізняється надійністю та елегантністю.

Примітка [33]: Елегантністю C# відрізняється через велику різноманітність синтаксичних конструкцій. А великої надійності було досягнуто за допомогою роботи CLR машини, котра на відміну від інших компіляторів запускає розроблений додаток на віртуальному процесорі. Тому у разі виникнення будь-яких помилок це ніяк не вплине на роботу інших програм у системі, але це також означає, що для запуску програми потрібен додатковий час. Відповідно програми написані мовою програмування C# надійніші, але менш швидкі (ніж самі програми написані на C++).

Оскільки в мові програмування C# є можливість використання та інтеграції із різними платформами і фреймворками, варто розгянути одну потужну модульну платформу, таку як .NET. Зазвичай C# та .NET потрібно розглядати разом, це два взаємопов'язаних продукта від Microsoft.

.NET (раніше відома як .NET Core) - це модульна платформа для розробки програмного забезпечення з відкритим вихідним кодом. Сумісна з операційними системами як Windows, Linux і macOS. Була випущена компанією Microsoft. Підтримує такі мови програмування: C#, Visual Basic .NET (частково) та F# [34].

Коли говорять C#, часто мають на увазі технології платформи .NET (Windows Forms, WPF, ASP.NET, Xamarin). І навпаки, коли говорять .NET, нерідко мають на увазі C#. Між собою ці поняття пов'язані, але ототожнювати їх не потрібно. C# було створено спеціально для роботи з фреймворком .NET, де саме поняття .NET дещо ширше. Фреймворк .NET представляє потужну платформу створення додатків.

Дана платформа часто оновлюється, вдосконалюється і забезпечує легку і потужну розробкудодатків.

За останього випуску 6 версії: .NET 6 поєднує пакет SDK, базові бібліотеки та середовище виконання для мобільних, класичних, хмарних програм та додатків Інтернету речей. Крім цього екосистема .NET 6 пропонує такі можливості [32] [35]:

- Підтримка кількох мов. Оскільки, основою платформи Common Language Runtime (CLR) – загальномовне середовище виконання, завдяки якому .NET при компіляції коду будь-якою з мов компілюється довершується загальною мовою CIL (Common Intermediate Language) - свого роду асемблер платформи .NET. Тому за певних умов ми можемо зробити окремі модулі однієї програми окремими мовами.
- Кросплатформність. .NET є платформою, що переноситься (з деякими обмеженнями) та розгортається на різних середовищах. Наприклад, остання версія платформи на даний момент – .NET 6 підтримується на більшості сучасних ОС Windows, MacOS, Linux. Використовуючи різні технології на платформі .NET, можна розробляти програми на мові C# для різних платформ - Windows, MacOS, Linux, Android, iOS, Tizen.
- Потужна бібліотека класів. .NET представляє єдину, підтримувану всіма мовами, бібліотеку. І який би додаток не було написано на C# - текстовий редактор, чат або складний веб-сайт - так чи інакше використовується бібліотеку класів .NET.
- Різноманітність технологій. Загальномовне середовище виконання CLR та базова бібліотека класів є основою цілого стеку технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних, у цьому стеку технологій, призначено технологію ADO.NET та Entity Framework Core. Для побудови графічних додатків з багатим насиченим інтерфейсом – технологія WPF та WinUI, для створення більш простих графічних

додатків – Windows Forms. Для розробки кросплатформових мобільних та десктопних програм - Xamarin/MAUI. Для створення веб-сайтів та веб-додатків - ASP.NET і т.д. До цього варто додати Blazor – новий фреймворк, що розвивається і набирає популяність. Він працює як .NET, але дозволяє створювати веб-додатки як на стороні сервера, так і на стороні клієнта.

- Максимальна продуктивність. NET 6 та Visual Studio 2022 надають можливості гарячого перезавантаження, пропонують нові засоби Git, інструменти інтелектуального редагування коду, надійні засоби діагностики, тестування та більш ефективної спільної роботи груп.
- Спрощена технологія: розпочати роботу дуже просто. Нові можливості мови C# 10 дозволяють скоротити обсяг створюваного коду. А інвестиції в веб-рішення та мінімальні API дозволяють швидко створювати невеликі та швидкодіючі мікрослужби.
- Висока рентабельність: .NET 6 - це найшвидша веб-платформа комплексної розробки, яка знижує витрати на обчислювальні ресурси при роботі в хмарі.

Також слід відзначити таку особливість мови C# і фреймворку .NET, як автоматичне складання сміття. Тобто, розробникам здебільшого не доводиться, думати про звільнення пам'яті, як це є у C++. Вищезгадане середовище CLR само викличе збирач сміття та очистить пам'ять.

Нововведень в .NET 6 куди більше ніж зазначено вище, їх краще зрозуміти і відчути уже на практиці, у використанні.

Тому, підсумовуючи, платформа .NET є досить конкурентним рішенням для сучасної веб-розробки, особливо серверної частини. Швидкодія середовища виконання, простота розробки, баланс між складністю мови та її можливостями

роблять .NET найкращим вибором, як для невеликих проектів, так і для великих підприємницьких-рішень.

2.5 Мікросервіси

У час коли технології розвиваються, а тенденції на додатки міняються майже не щодня, потрібно бути у русі течії. Оскільки монолітні програми є важкими у розширенні та дороговартісними у постійній підтримці функціоналу, тому розробники все активніше переходять на мікросервісну архітектуру.

Моноліт будуються як єдине ціле. Будь які зміни, навіть самі невеликі, потребують перебудови та розгортання всього додатку. З часом стає складніше зберігати хорошу модульну структуру, зміни логіки одного модуля мають тенденцію впливати на код інших модулів. Монолітні програми також може бути важко масштабувати, коли різні модулі мають конфліктні вимоги до ресурсів. Наприклад, один модуль може реалізовувати логіку обробки зображень з інтенсивним використанням процесору. Інший модуль може бути вимогливим до використання оперативної пам’яті. Однак, оскільки ці модулі розгортаються разом, доведеться йти на компроміс з вибором апаратного забезпечення. Масштабувати доводиться весь додаток, навіть якщо це потрібно тільки для одного модуля цього додатку [36]. Щодо підтримки монолітної програми, то використання важких та дороговартісних сервісів, обтяжують ресурси продукту, команди та проекту, а також вартують не дешево.

Термін “Мікросервісна архітектура”, також відомий як мікросервіси, з’явився в середині 2010-х, щоб описати особливий архітектурний стиль розробки програмних додатків. Цей стиль розробки отримав розповсюдження в зв’язку з розвитком практик гнучкої розробки та DevOps [36].

Архітектурний стиль мікросервісів — це підхід, коли єдиний додаток будується як сукупність невеликих, самодостатніх, незалежних, не тісно зв'язаних сервісів, що спілкуються між собою за допомогою легких механізмів як то HTTP, gRPC, AMQP. Ці сервіси побудовані навколо бізнес-потреб (кожен відповідальний за конкретний процес) та розгортаються незалежно з використанням повністю автоматизованого середовища. Існує абсолютний мінімум централізованого управління цими сервісами. Самі по собі сервіси можуть бути написані на різних мовах і використовувати різні технології зберігання даних [36].

Одна з причин використання мікросервісів полягає в тому, що компанії хочуть мати можливість швидко щось змінювати, щоб швидше реагувати на зміни бізнес-вимог, випереджати конкурентів. Мікросервіси допомагають розробникам доставляти зміни швидше, безпечніше і з більш високою якістю, тобто зберігати швидкість розвитку продукту, навіть коли той стає неосяжних розмірів. Адже не тісно зв'язані сервіси дають можливість проводити зміни з більшою частотою ітерацій мінімізуючи вплив змін на решту частин системи[36].

Варто зазначити, що мікросервісна архітектура C# — гарний варіант для мікросервісної архітектури завдяки інтегрованому середовищу розробки Visual Studio, підтримці Microsoft і зручності розробки ПЗ для Windows.

ASP.NET Core — фреймворк для створення кросплатформних мікросервісів. Його переваги у простому розгортанні в хмарі та різних операційних системах — Windows, Linux та macOS. Ще одна причина вибрати цей фреймворк — він став одним із перших фреймворків, у якому з'явилися рішення на основі архітектури мікросервісів[37].

Коли йдеться про забезпечення гнучкої розробки і доставки складних корпоративних додатків — такий спосіб розробки має значні переваги[36]:

- Сервіси запускається швидше, що робить розробників більш продуктивними та прискорює розгортання.
- Кожен сервіс може бути розгорнутий незалежно від інших. Тож якщо ви змінюєте щось в одному з них, ви можете розгорнути ці зміни, не чіпаючи інших мікросервісів, які можуть продовжувати працювати.
- Кожен сервіс може бути масштабований окремо.
- Баг в одному мікросервісі не підірве роботу системи. Неполадки у мікросервісі не повинні зламати весь додаток. Швидше за все, вони не вплинуть суттєво на роботу додатку, особливо великого.
- Усуваються будь-які довгострокові зобов'язання щодо технології. При розробці нового сервісу можна вибрати новий стек технологій. Будь-який сервіс у системі можна замінити. Його можна переписати з нуля в межах прийнятного часу та бюджету без необхідності перебудовувати всю систему.
- Мікросервіси, як правило, краще організовані, оскільки кожен мікросервіс має дуже специфічну роботу і не займається роботою інших сервісів. Тож потенційно легші для розуміння, підтримки і тестування.
- Також відокремлені сервіси легше перекомпонувати і переналаштувати, щоб виконувати задачі різних додатків (наприклад, обслуговувати веб-клієнтів і публічний API).

Недоліки мікросервісів[36]:

- Розробники повинні мати справу з додатковою складністю створення розподіленої системи.
- Складність розгортання. У виробництві існує також оперативна складність розгортання та управління системою, що складається з багатьох різних типів послуг.

- При будуванні нової архітектури мікросервісу, ймовірно, знайдеься багато багатопланових проблем, які не очікувались під час розробки.

Залишаючись монолітним в сучасному світі, важко змінюватись разом з потребами примхливого ринку. Ось чому створення програмного забезпечення у вигляді мікросервісів шалено набирає популярність на противагу монолітному підходу[38].

2.6 Службова шина та запланована доставка повідомлень (Azure Service Bus and Scheduled Messages)

Досить часто виникає необхідність зв'язувати разом кілька різних систем, сервісів або забезпечити їх синхронізацію. Звичайно, випадки бувають різні й унікальні, однак опис власного рішення, як правило, досить затратно, як за часом, так і за ресурсами[39].

Дані передаються між різними програмами та службами за допомогою повідомлень.

Повідомлення — це контейнер з вмістом певних даних, прикрашений метаданими. Де даними може бути будь-яка інформація, включно зі структурованими даними, закодованими в таких поширених форматах, як JSON, XML, Apache Avro, Plain Text [40].

Azure Service Bus — це повністю керований корпоративний брокер повідомлень із чергами повідомлень і темами публікації та підписки (у просторі імен). Службова шина використовується для відокремлення програм і служб одна від одної, надаючи такі переваги [40]:

- Робота з балансування навантаження між конкуруючими працівниками
- Безпечна маршрутизація та передача даних і контроль через межі служби та програми

- Координація транзакційної роботи, яка вимагає високого ступеня надійності

Виділяють основні сценарії обміну повідомденнями [40]:

- Поділ завдань між додатками.
- Балансування навантаження.
- Розділи та передплати.
- Транзакції.
- Виділення сеансів обміну повідомленнями.

Оскільки Службова шина надається за моделлю PaaS (платформа як послуга), вона дозволяє не турбуватися про наведені нижче дії. які виконує платформа Azure [40]:

- обробка апаратних збоїв;
- встановлення виправлень для операційних систем чи продуктів;
- розміщення журналів та управління дисковим простором;
- керування резервними копіями;
- відпрацювання відмови на резервний комп'ютер;

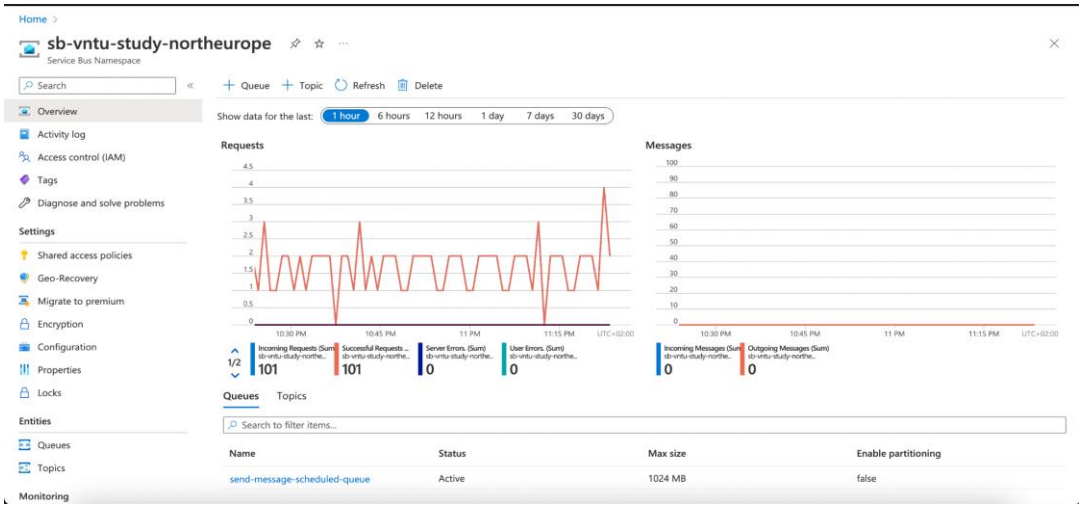


Рисунок 2.4 - Зображення інтерфейсу Azure Service Bus контейнеру всіх компонентів

Azure Service Bus виділяється серед інших брокерів повідомлень запаморочливим набором додаткових і корисних функцій, які він надає відразу та безкоштовно.

Однією з таких зручних функцій є можливість планування повідомлень у вашій черзі задач (Scheduled Messages). Припустимо, наприклад, що ви організатор події та хочете повідомити людей за кілька днів до події. Ця функція дає вказівку Service Bus просто надіслати повідомлення в певний момент часу [42].

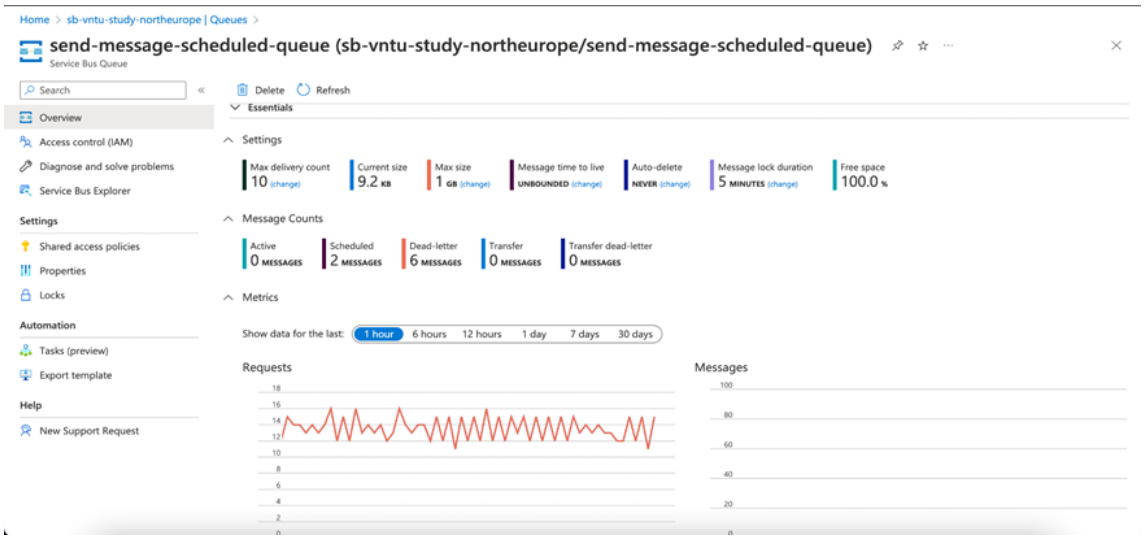


Рисунок 2.5 - Зображення інтерфейсу Azure Service Bus Scheduled Messages Queue

Послідовність і відмітка часу — це дві функції, які завжди ввімкнено в усіх об’єктах службової шини та проглядаються через властивості SequenceNumber та EnqueuedTimeUtc отриманих або переглянутих повідомлень.

У тих випадках, коли абсолютний порядок повідомлень є значущим і/або коли споживачеві потрібен надійний унікальний ідентифікатор для повідомлень, брокер позначає повідомлення без пропусків, зростаючим порядковим номером відносно черги або теми. Для розділених об’єктів порядковий номер видається відносно розділу [41].

Значення SequenceNumber — це унікальне 64-розрядне ціле число, яке присвоюється повідомленню, оскільки воно приймається та зберігається брокером і функціонує як його внутрішній ідентифікатор. Для розділених об’єктів верхні 16 бітів відображають ідентифікатор розділу. Послідовні номери повертаються до нуля, коли діапазон 48/64-біт вичерпується [повід. мс док].

Рекомендується десь зберігати повернений 'sequenceNumber'. Це потрібно на випадок, якщо буде потрібно видалити заплановане повідомлення. Видалення або, точніше, скасування запланованого повідомлення можна реалізувати, як у наступному прикладі коду [43].

Можливість відміток часу діє як нейтральний і надійний орган, який точно фіксує час надходження повідомлення за UTC, що відображається у властивості EnqueuedTimeUtc . Значення корисне, якщо бізнес-сценарій залежить від кінцевих термінів, наприклад, чи було подано робочий елемент у певну дату до півночі [41].

2.7 Blazor framework - клієнтська частина на мові C#

Blazor представляє UI-фреймворк для створення інтерактивних програм, які можуть працювати як на стороні сервера, так і на стороні клієнта, використовуючи платформу .NET.

У своєму розвитку фреймворк Blazor зазнав великого впливу сучасних фреймворків на створення клієнтських додатків - Angular, React, VueJS. Зокрема, це проявляється в ролі компонентів при побудові інтерфейсу користувача. У ході розробки обох частин (клієнтської та серверної) використовується мова програмування C#, замість JavaScript. А для опису візуального інтерфейсу використовуються стандартні HTML та CSS [44].

Фреймворк Blazor розвивається як opensource-проект, вихідний код якого можна знайти в репозиторії на github: https://github.com/dotnet/aspnetcore/tree/master/src/Components. [44]

Blazor надає розробникам такі переваги [44]:

- Написання коду веб-застосунків за допомогою C# замість JavaScript.
- Використання можливостей екосистеми .NET, зокрема бібліотек .NET при створенні додатків, безпеки та продуктивності платформи .NET.
- Клієнтська та серверна частини програми можуть використовувати загальну логіку.
- Використання Visual Studio як інструмент для розробки, який має вбудовані шаблони для спрощення створення програми.

Функціонально на даний момент Blazor поділяється на дві підсистеми [44]:

- Blazor Server: дозволяє створювати серверні програми та підтримується ASP.NET Core
- Blazor WebAssembly: дозволяє створювати односторінкові інтерактивні програми клієнтської сторони, які запускаються у браузері користувача та працюють за допомогою технології WebAssembly.

У ході розробки “Веб порталу викладача” є необхідним і достатнім використання Blazor WebAssembly для односторінкової інтерактивної програми, не знаючи в повній мірі Angular, React чи VueJS.

Однією з переваг Blazor WebAssembly є те, що він може оптимізувати завантаження. Зокрема, при розготанні програми код, що не використовується, забирається лінкером (компонувальником) IL (Intermediate Language). Крім того, всі необхідні файли середовища виконання .NET і завантаження кешуються в браузері.

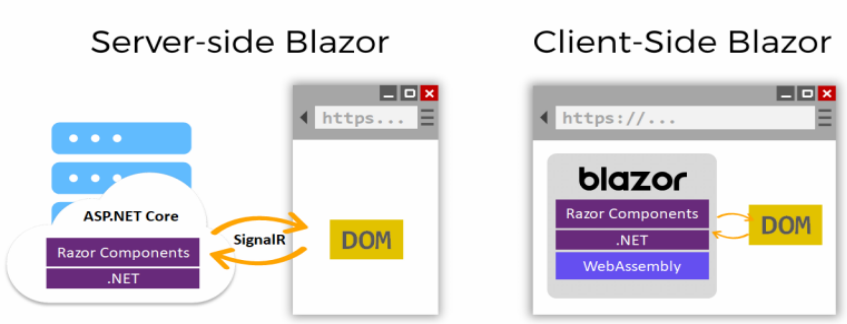


Рисунок 2.6 - Схема роботи функціоналу на Blazor

Blazor WebAssembly не залежить від сервера. За великим рахунком, нам може бути достатньо статичного сервера, на якому розміщені всі файли програми. Всі необхідні файли завантажуються браузером, і після завантаження файлів програма працює повністю на стороні браузера і не залежить від сервера [44].

У той же час Blazor WebAssembly має низку обмежень. Наприклад, браузер повинен підтримувати технологію WebAssembly – на даний момент останні версії поширених браузерів (Google Chrome, Mozilla Firefox, Opera, Microsoft Edge, Yandex Browser) підтримують цю технологію. Також браузеру необхідно завантажити файли великого розміру, оскільки програма повністю відпрацьовує на стороні клієнта, що збільшує навантаження на мережу та час завантаження. Ну і крім того, в цьому випадку можливості програми обмежені браузером, в якому програма запускається [44].

Рішення з використання фреймворку Blazor покриває всі потреби у web-додатках, як на стороні сервера, так і клієнта. Тим більше, що компанія Microsoft не планує на цьому зупинятися і планує розвивати продукт. Ймовірно, що Blazor незабаром можна буде використовувати як для мобільних, так і десктопних додатків.

2.8 Система моніторінгу Azure Application Insights

Найцінніший ресурс в наш час – це інформація, тому так багато сил і коштів компанії витрачають на збір різних даних. Але інформацію треба не тільки зібрати, її треба ще й проаналізувати [45].

Коли даних мало, то і проблеми в аналізі немає. Якщо інформації стає все більше, то рішення, прийняте на основі її аналізу, буде з досить великою часткою ймовірності правильним. Але ось тут постає питання в аналізі[45].

Сьогодні існує безліч систем для візуалізації зібраних даних, їх аналізу, досліджень та порівнянь. Найбільшої популярності набирають мульти функціональні і прості системи такі як: Kibana, Grafana, Azure Application Insights.

Kibana була створена на основі стеку Elasticsearch, відомого завдяки аналізу та керуванню журналами. Для порівняння, Grafana була створена в основному для моніторингу метрик, що підтримує візуалізацію для баз даних часових рядів [46].

На відміну від популярних Kibana та Grafana, Azure Application Insights уже йде інтегрованим у Azure portal, що не потребує інтеграції та лишніх дій із підключенням, що насправді полегшує аналіз даних по розроблюваному продукту. До того ж Insights не потребуюють великого ресурсу, як час і кошти, плата стгується лише за час і кількість задіяних сервісів у ході використання, що є продуктивним та економічним рішеням.

Application Insights є розширенням Azure Monitor і забезпечує функції моніторингу продуктивності програми. Інструменти APM корисні для моніторингу додатків від розробки, тестування до виробництва такими засобами[47]:

- Змога заздалегідь визначити якість процесу роботи програми.
- Можливість реактивно переглядати дані про виконання програми, щоб визначити причину інциденту.

На додаток до збору показників і даних телеметрії програми, які описують діяльність і стан програми, Application Insights також можна використовувати для збору та зберігання даних журналу трасування програми.

Журнал трасування пов’язується з іншими телеметричними даними, щоб надати детальне уявлення про діяльність. Для додавання журналу трасування до існуючих програм потрібно лише вказати місце призначення для журналів; структуру записів рідко потрібно змінювати.

Application Insights надає інші не обмежені функції[47]:

- Метрики в реальному часі – спостерігайте за активністю вашої розгорнутої програми в режимі реального часу без впливу на середовище хоста.
- Доступність – також відомий як «синтетичний моніторинг транзакцій», досліджуйте зовнішні кінцеві точки додатків, щоб перевірити загальну доступність і швидкість реагування з часом.
- Інтеграція GitHub або Azure DevOps – створюйте робочі елементи GitHub або Azure DevOps у контексті даних Application Insights.
- Легке використання – журнали логів покажуть популярні серед користувачів функції та взаємодію програм.
- Smart Detection – автоматичне виявлення збоїв і відхилень за допомогою проактивного телеметричного аналізу.

Крім того, Application Insights підтримує розподілене трасування, а також розподілену кореляцію компонентів. Ця функція дозволяє шукати та візуалізувати наскрізний потік певного виконання чи транзакції. Здатність наскрізного відстеження активності стає все більш важливою для програм, створених як розподілені компоненти або мікросервіси[47].

Карта додатків дозволяє високорівневий перегляд архітектури додатків зверху вниз, а також миттєві візуальні посилання на стан і швидкість реагування компонентів.

Application Insights вмикаються за допомогою автоматичного інструментарію або шляхом додавання пакета SDK Application Insights до коду програми. Тут підтримується багато мов програмування, розташування програм може бути на Azure, локально або в іншій хмарі[47].

Агент Application Insights або SDK попередньо обробляє телеметрію та показники, перш ніж надсилати дані в Azure, де вони надходять і обробляються, перед збереженням у журналах моніторингу Azure (Log Analytics).

Найпростіший спосіб почати використовувати аналітику програми — через портал Azure, де вбудовані візуальні засоби. Досвідчені користувачі можуть запитувати базові дані безпосередньо для створення користувацьких візуалізацій за допомогою інформаційних панелей і робочих книг Azure Monitor [47].

Режим перегляду «Помилка » – показує, які компоненти чи дії викликають збої, помилки сортування та винятки. Вбудовані представлення корисні для проактивного відстеження працездатності додатків і реактивного аналізу першопричини помилок.

Сповіщення Azure Monitor, повідомляють про потенційні проблеми, якщо ваша програма або компоненти відхиляються від встановленого базового рівня.

Ціни Application Insights базуються на споживанні; ви платите лише за те, що використовуєте. Щоб отримати додаткові відомості про ціни, доступна сторінка Ціни Azure Monitor та оптимізація витрат .

Автоматичний інструментарій є кращим методом інструментування. Він не вимагає інвестицій розробника та усуває майбутні накладні витрати, пов’язані з

оновленням SDK . Це також єдиний спосіб інструментування програми, у якій ви не маєте доступу до вихідного коду[48].

Щоб використовувати SDK, ви встановлюєте невеликий пакет інструментів у свою програму, а потім інструментуєте веб-програму, будь-які фонові компоненти та JavaScript на веб-сторінках. Програма та її компоненти не обов'язково розміщуються в Azure. Інструменти контролюють вашу програму та спрямовують дані телеметрії до ресурсу Application Insights за допомогою унікального маркера. Вплив на продуктивність вашої програми невеликий; виклики відстеження не блокуються та групуються для надсилання в окремому потоці[47].

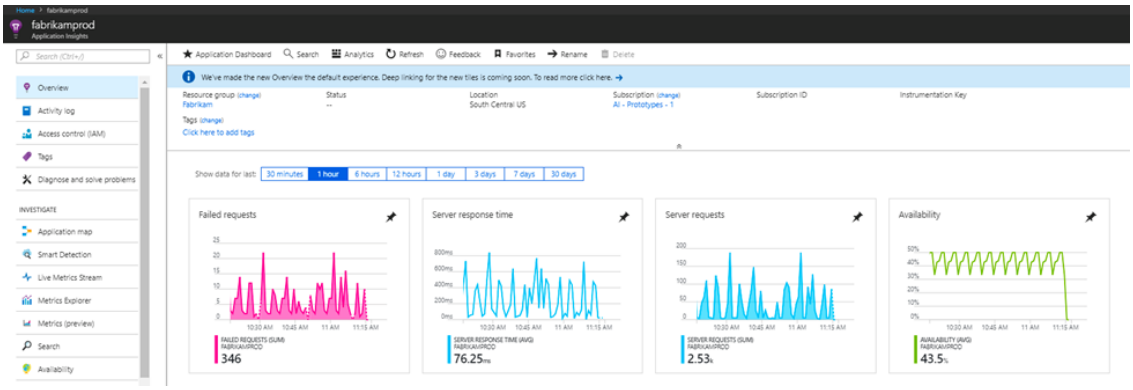


Рисунок 2.7 - Зображення інтерфейсу оглядової панелі Application Insights

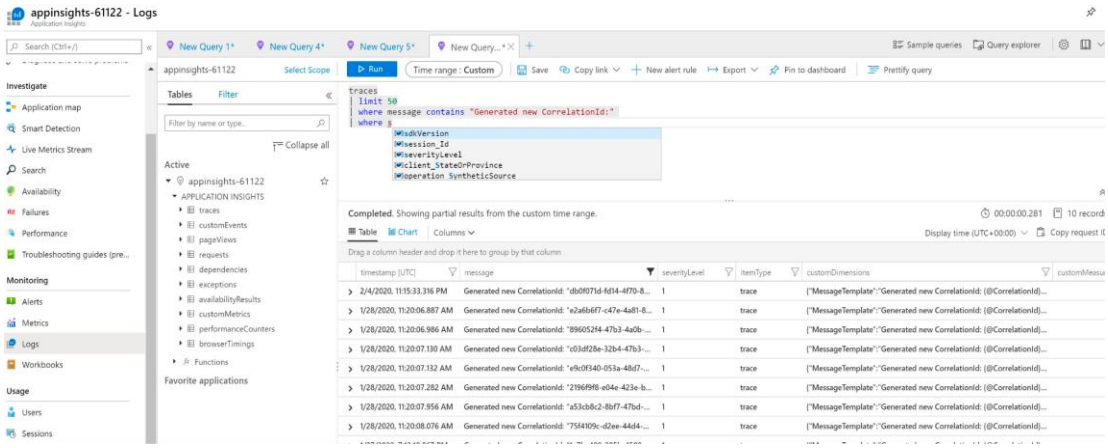


Рисунок 2.8 - Зображення інтерфейсу журналу логування Application Insights

Моніторингова система є пов'язною з реальним часом і будь-які зміни записуються і відображаються автоматично.

Використовуючи Application Insights у ході даної роботи отримані результати опитувань, контрольних робіт чи тестування будуть представлені у стаститичному вигляді, наприклад:

- Дата та час запису;
- ПІБ студента;
- кількість спроб проходження тесту;
- середній бал/оцінка;
- група, тощо.

2.9 Система відправки електронних повідомлень - SendGrid

Аналізуючи системи відправки електронних повідомлень, таких як: SurveyMonkey - платформа для створення та проведення онлайн-опитування, доступна до інтегрування, але вибір і створення шаблонних форм є обмеженою у безкоштовному пакеті; чи Google Forms - це програмне забезпечення для адміністрування опитувань, яке входить до складу безкоштовного пакету веб-редактора Google Docs Editors, пропонованого Google. Де дане ПЗ є повністю безкоштовним та доступним у створенні будь-яких форм для цілі вирішення маркетингових задач, але не є доступним у інтегруванні.

Однак жодна система із представлених не має можливості автоматизації. Вирішуючи саме проблему автоматизації навчання через відправку електронних повідомлень, бо вирішено використати SendGrid.

Його функціонал та надані інструменти доступні до автоматизації та інтегрування з такими сервісами: Microsoft Azure cloud, що має безліч різних додатків, сервісів та функцій для автоматизації, а також із Microsoft Outlook - поштовим сервісом.

SendGrid – це платформа email-маркетингу, яка допомагає бізнесу автоматизувати поштові розсилки та пропонує цілу низку інструментів для цього. Користувачі можуть створювати маркетингові розсилки з нуля (використовуючи drag and drop або HTML-конструктор), так і на основі численних адаптивних шаблонів, доступних у бібліотеці сервісу [48].

Ще однією з ключових послуг SendGrid є набір інструментів для автоматизації процесу електронної пошти. Він дозволяє відправляти листи, що повторюються, складати графік розсилки серій листів, підтримує автоматичне відправлення листів по тригерах і ряд інших корисних опцій. Також тут представлено низку додаткових послуг, включаючи виділену IP-адресу, тестування ефективності розсилок, управління списками, аналітика та статистика розсилок, форми для залучення нових передплатників тощо [48].

Окремим інструментом SendGrid виступає Email API для розробників (https://sendgrid.com/solutions/email-api/), що дозволяє гнучко інтегрувати функціонал сервісу з будь-яким зовнішнім програмним забезпеченням (сайтами, десктоп-програмами, мобільними додатками тощо). Перелік інструментів включає RESTful API, SMTP, бібліотеки для підтримки різних мов програмування (Ruby, Node.JS, Python, Go, Java, C#), а також комплект інтерактивної документації [48].

Інтеграція стороннього ПЗ з SendGrid через Email API робить використання сервісу більш зручним та ефективним. Вона дає можливість створювати нові шаблони розсилок у інтерфейсі користувача і відправляти їх програмними методами. Також тут доступна конфігурація налаштувань облікового запису (дозволи, сегментація тощо), оперативна діагностика помилок, аналіз продуктивності всіх учасників команди тощо [48].

Доступні через API опції безпеки включають керування рівнями доступу, API-ключі, 2FA-автентифікацію, керування доступом до IP-адрес, TLS-

шифрування, захист повідомлень стандартами автентифікації DMARC/DKIM, систему захисту даних Event Webhook і т.д [48].

Отже, основні характеристи SendGrid [49]:

- SMTP-сервіс.
- Кастомні інтеграції API.
- Відстеження відкриттів та кліків.
- Шаблони повідомлень.
- Відмова від відстеження.
- Репутація моніторингу.
- Управління списками.
- Виділені IP-адреси.
- SMTP API.
- Моніторинг ISP.
- Тестування фільтру.
- Балансування навантаження.
- Аналіз Webhook.
- Зворотній зв'язок.
- DKIM Налаштування.
- SMTP Relay.
- Підтримка 24/7.

SendGrid – це платформа, яка покращить транзакційні розсилки та дозволить масштабувати їх для вирішення задач email-маркетингу. Вона пропонує гнучкі веб та SMTP API-інтерфейси, може легко інтегруватися з будь-якою хмарною інфраструктурою. SendGrid підтримує різні фреймворки, мови та програми [49]. А інтеграція із Microsoft Azure Function дозволить автоматизувати автоматичне розсилання повідомлень, виконуючи розсилку лише тоді, коли це необхідно

користувачеві, не переплачуючи за переліміт у кількості листів в день, та додати більше корисних та економних функцій.

2.10 Хмарне рішення Azure devops

Будь яка основа сучасного проекту є місцем, де можна відслідковувати, створювати нові задачі відносно проекту, відслідковування статус у цих процесів та створення загальних звітів щодо статусу. В середовищі Azure існує сервіс під назвою Azure Devops, який не тільки покриває вище сказані задачі, але і допомагає автоматизувати доставку кінцевого коду до користувача, створення кроків для тестування та зберігання коду проекту зі своїм внутрішнім git-репозиторієм [50].

Git — розподілена система керування версіями файлів та спільної роботи розробників. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, які базуються на відгалуженні і злитті гілок. Система спроектована як набір програм, спеціально розроблених з врахуванням їхнього використання у скриптах. Це дозволяє зручно створювати спеціалізовані системи керування версіями на базі Git або користувацькі інтерфейси[51].

Azure DevOps (раніше був Team Foundation Server (TFS) та Visual Studio Team Services) – продукт корпорації Microsoft, комплексне рішення, що об'єднує в собі систему керування версіями, збір даних, побудову звітів, відстеження статусів та змін по проекту, тестування та призначений для спільної роботи над проектами з розробки програмного забезпечення. Цей продукт доступний як у вигляді окремого програмного засобу, так і у вигляді платформи для Visual Studio team System (VSTS).

Azure DevOps доступний у двох видах: на власних серверах («Server») і онлайн («Servises»). Хмарні сервіси розташовані на хмарній платформі Microsoft Azure. Користувач входить використовуючи обліковий запис Microsoft, щоб налаштувати оточення, створюючи проекти і додаючи нових членів команд [52].

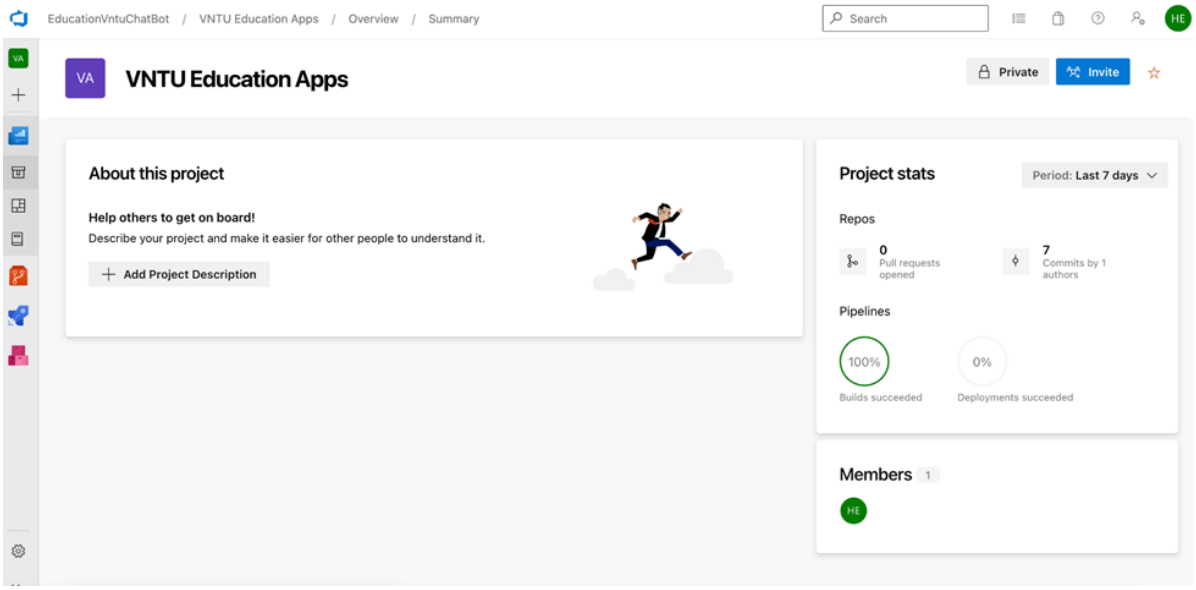


Рисунок 2.8 – Зображення робочого середовища Azure DevOps

Головними компонентами Azure DevOps є Repositories, Pipelines, DashBoard [50].

Секція Repositories містить всю інформацію по версії коду - тобто версія коду в відповідній вітці Git, список віток гіт, список тегів гіт і т.д. В даному розділі розміщується код, який відповідає за розгортання середовища для боту, та за логіку боту в середині сервісів [50].

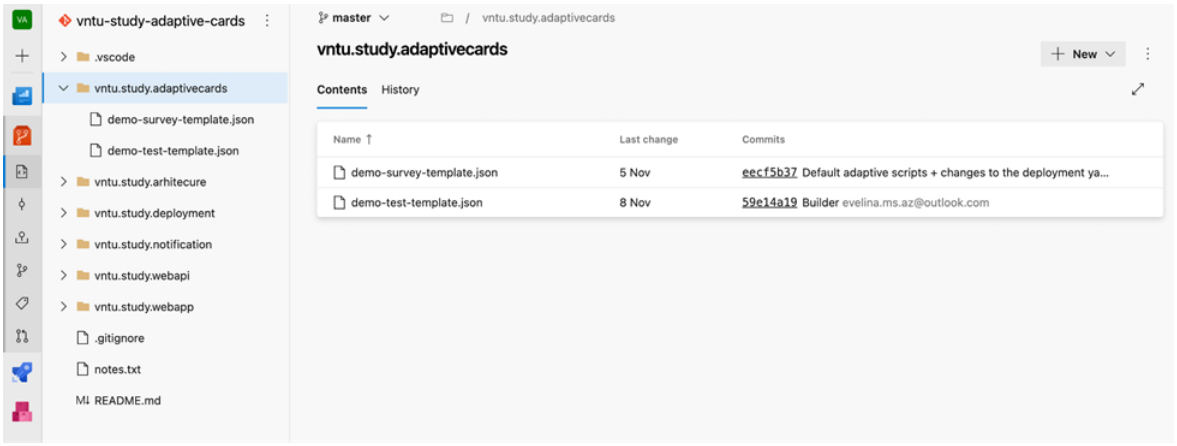


Рисунок 2.9 – Зображення робочого середовища Azure DevOps. Компонент Repositories

Секція Pipelines відповідає за CI/CD, тобто надає служби збірки і випуску для підтримки безперервної інтеграції і доставки додатків [50].

CI / CD – це концепція безперервної інтеграції коду (continuous integration) та безперервного його розгортання (continuous delivery) підчас розробки певного програмного забезпечення, що прискорює збірку, тестування та розгортання додатків та програм. Сьогодні програмісти використовують CI / CD технологію для будь яких задач, вона необхідна для розробки програмного забезпечення із застосуванням Agile-методології, для швидкого налагодження робочого програмного забезпечення [50].

За допомогою процесу CI/CD було реалізовано автоматичне збирання коду (артефактів проекту), передавання артефактів на частину доставки коду, розгортання нового середовища для чат боту та оновлення коду на розгорнутих сервісі. Таким чином будь які зміни в коді автоматично визначаються частиною CI і завантажуються в сервіси хмарного середовища. Підчас CD – доставки, зміни автоматично з’являються і відображаються на сайті за 10-20 хвилин, без потреб диспетчера чи розробника у слідкуванні за процесом [50].

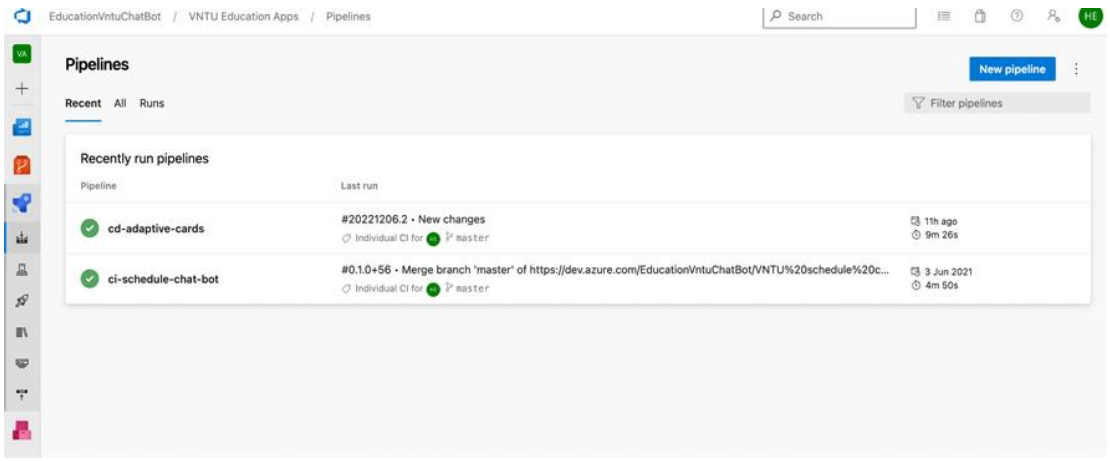


Рисунок 2.10 – Зображення робочого середовища Azure DevOps – Pipelines (CI)

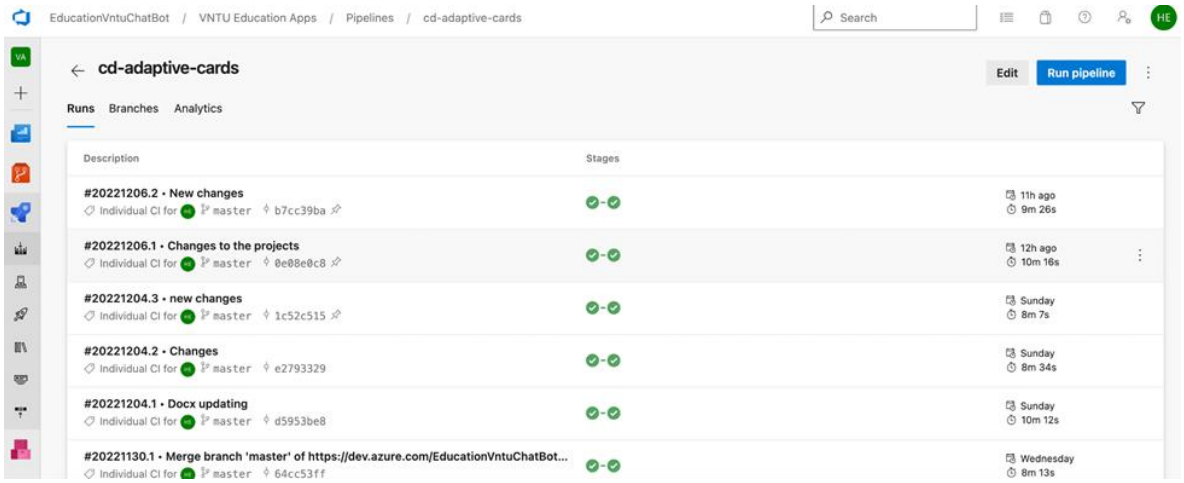


Рисунок 2.11 – Зображення робочого середовища Azure DevOps – Pipelines (CD)

Azure Boards надає набір гнучких засобів для підтримки планування і відстеження роботи, дефектів коду і проблем з використанням методів Kanban і Scrum [50].

Для автоматизації процесу був використаний Powershell script, який автоматизує процес створення ресурсів і всіх додаткових сервісів і компонентів для середовища розробки [50].

PowerShell - це міжплатформне рішення для автоматизації завдань, яке складається з оболонки командного рядка, мови сценаріїв та структури управління конфігурацією. PowerShell працює на Windows, Linux та macOS.

Shell - це сучасна командна оболонка, яка включає найкращі можливості інших популярних оболонок. На відміну від більшості, які приймають і повертають лише текст, PowerShell приймає та повертає об'єкти .NET. Оболонка включає такі особливості [53]:

- Надійна історія командного рядка;
- Завершення вкладки та передбачення команд;
- Підтримує псевдоніми команд і параметрів;
- Трубопровід для ланцюгових команд;
- Вбудована довідкова система, подібна до робочих сторінок Unix.

Як мова сценаріїв, PowerShell зазвичай використовується для автоматизації управління системами. Він також використовується для побудови, тестування та розгортання рішень, часто в середовищі CI / CD. PowerShell побудований на .NET Common Language Runtime (CLR). Всі входи та виходи є .NET-об'єктами. Не потрібно аналізувати текстовий висновок для вилучення інформації з вихідних даних. Мова сценаріїв PowerShell містить такі функції[53]:

- Розширюється за допомогою функцій, класів, сценаріїв та модулів.
- Розширювальна система форматування для легкого виведення.
- Розширювальна система типів для створення динамічних типів.
- Вбудована підтримка поширених форматів даних, таких як CSV, JSON та XML

Конфігурація бажаного стану PowerShell (DSC) - це система управління в PowerShell, яка дозволяє управляти корпоративною інфраструктурою за допомогою конфігурації як коду. За допомогою DSC ви можете [53]:

- Створювати декларативні конфігурації та власні сценарії для повторюваних розгортань.
- Застосовувати налаштування конфігурації та звітувати про відхилення конфігурації.
- Розгортати конфігурації за допомогою моделей push або pull.

Для створення середовища і всіх ресурсів використовується Azure Resource Templates, які містять всю інформацію по структурі ресурс групи і всіх проектах. Таким чином можна без проблем відновити всі ресурси в разі їх пошкодження чи видалення [50].

Azure Arm (Azure Resource Manager) templates – це шаблон у форматі JSON, який містить зрозумілу розмітку для платформи Azure та описує всі ресурси середовища. Використовуючи дані ресурси, описуються такі сервіси як Storage Account та сервіси для логування. Кожний сервіс має свою JSON структуру, яка визначена документацією Майкрософта, більш детально описується сервісний план, назва самого сервісу і його розміщення [50].

Сервісний план відповідає за потужність, за використовувану пам’ять та ціну. Розміщення – регіон в якому знаходиться сервіс (Східна Європа), локація визначається Azure автоматично [50].

Прийшовши дані кроки завантажується та створюється середовище розробки. Кожний крок містить інформацію розгортання, що дозволяє визначити будь які помилки на кожному етапі. Завершене розгортання маркується зеленим кольором [50].

2.11 Система описування інфраструктури – bicep

Bicep – це мова шаблону Resource Manager, яка використовується для декларативного розгортання ресурсів Azure. Bicep - це мова DSL, тобто вона призначена для конкретного сценарію або "домена". Вона не призначена для використання як стандартної мови програмування для написання програм. Bicep використовується лише для створення шаблонів Resource Manager. Вона проста для розуміння та освоєння, незалежно від досвіду роботи з іншими мовами. У шаблонах Bicep допускаються всі типи ресурсів, версії API та властивості[54].

Якщо розробник раніше вивчав для використання шаблонів мову JSON, то помітить, що Bicep спрощує процес самого створення шаблону. Вона надає більш простий для розуміння синтаксис, кращу підтримку модульності та багаторазово використовуваного коду, а також підвищену безпеку типу. Для створення шаблону ARM за допомогою JSON потрібні складні вирази, а остаточний результат може бути докладним [54].

Для розробки шаблонів Bicep надає безліч покращень порівняно з JSON, включаючи наведені нижче [54]:

- Простіший синтаксис. Bicep надає простіший синтаксис для написання шаблонів. Є можливість безпосередньо посилатися на параметри та змінні без використання складних функцій. Щоб об'єднати значення імен та інших елементів, використовується інтерполяція рядків замість об'єднання. Також можна посилатися на властивості ресурсу безпосередньо, використовуючи його символьне ім'я замість складних інструкцій. Ці покращення синтаксису допомагають і в розробці, і читанні шаблонів Bicep.
- Модулі. Розробники можуть розбивати складні розгортання шаблонів на невеликі файли модулів та посилатися на них в основному шаблоні. Ці модулі забезпечують більш просте керування та більшу можливість

повторного використання. Можливо навіть поділитися своїми модулями зі своєю командою.

- Автоматичне керування залежностями. У більшості випадків Bicep автоматично виявляє залежність між ресурсами. Цей процес усуває частину роботи, пов'язану з розробкою шаблонів.
- Перевірка типу та IntelliSense. Розширення Bicep для функцій Visual Studio Code має широкі можливості перевірки IntelliSense для всіх визначень API типів ресурсів Azure. Ця функція полегшує процес розробки.

Однак, серед недоліків роботи з Bicep є те, що файл Bicep не можна згенерувати з порталу Azure, оскільки під час декомпіляції шаблону ARM в Bicep функції можуть бути передані некоректно і спричинити потребу їх переписати [55].

Розглянемо наступний приклад шаблону Bicep, який визначає обліковий запис зберігання. Шаблон автоматично створює ім'я облікового запису зберігання. Після розгортання ідентифікатор ресурсу повертається як вихідні дані користувачеві, який виконує шаблон [54].

Bicep

```
param location string = resourceGroup().location
param namePrefix string = 'storage'

var storageAccountName = '${namePrefix}${uniqueString(resourceGroup().id)}'
var storageAccountSku = 'Standard_RAGRS'

resource storageAccount 'Microsoft.Storage/storageAccounts@2019-06-01' = {
  name: storageAccountName
  location: location
  kind: 'StorageV2'
  sku: {
    name: storageAccountSku
  }
  properties: {
    accessTier: 'Hot'
    supportsHttpsTrafficOnly: true
  }
}

output storageAccountId string = storageAccount.id
```

Рисунок 2.12 - Зображення коду шаблону розгортання на мові Bicep

При розгортанні ресурсу або ряду ресурсів в Azure розробник відправляєте шаблон Bicep до служби Resource Manager, яка все ще вимагає шаблони JSON. Кошти, вбудовані в Bicep, перетворюють шаблон Bicep на шаблон JSON. Цей процес, коли шаблон ARM обробляється як проміжна мова, називається транспіляцією. Перетворення відбувається автоматично під час надсилання розгортання або його можна виконати вручну [56].

Зображення різниці коду мовою Bicep та JSON [56]. Шаблон ліворуч — шаблон Bicep. Шаблон праворуч є шаблоном JSON.

Рисунок 2.13 - Зображення різниці коду мовою Bicep та JSON

Останні версії Azure CLI та модуля Azure PowerShell мають вбудовану підтримку Bicep. Для розгортання шаблонів Bicep і JSON можна використовувати ті самі команди розгортання. Наприклад, команда “az deployment group create --template-file ./main.bicep --resource-group storage-resource-group” розгортає шаблон Bicep у групі ресурсів під назвою “storage-resource-group” [56].

Після надсилання цього розгортання Resource Manager переглядає, що вже розгорнуто в Azure. Потім вона переглядає, що розробник намагається розгорнути

і налаштовує послідовність дій для досягнення цього стану. Всі ці дії передбачають виклик API Resource Manager.

Ви можете переглянути шаблон JSON, надісланий до Resource Manager за допомогою команди “bicep build” - “ bicep build ./main.bicep” [56].

Також однією з переваг використання Bicep є те, що створену інфраструктуру можна розгортати декілька разів протягом усього життєвого циклу розробки [55].

2.12 Висновки

У ході дослідження та аналізу даного розділу, було представлено і вибрано середовище для розробки інтерактивних електронних поштових повідомлень, визначено проблеми і потреби користувачів. Описано та проаналізовано використання новітніх технологій, взято до уваги їх переваги та недоліки.

3 РОЗРОБКА СИСТЕМИ АВТОМАТИЗАЦІЇ НАВЧАЛЬНОГО ПРОЦЕСУ

За даними результатами дослідження пропонується розробити інтерактивні форми в поштових електронних повідомленнях для автоматизації навчального процесу.

Перелік основних функцій поставленої задачі:

1. Можливість створювати і змінювати форми опитування.
2. Можливість встановлювати час відправки.
3. Функція відправки електронних повідомлень.
4. Відображення статистики результатів.
5. Автоматичне розгортування системи.
6. Система авторизації на базі Microsoft Identity.

Розробка складається з сервісів, які в свою чергу відповідають виключно за виконання задач одного направлення.

Клієнтська частина, що доступна для користувачів побудована з використанням сучасних технологій Microsofot - Blazor, WebAssembly.

Серверна частина для клієнтського додатку, яка відповідає за обробку всіх вхідних запитів і відправки повідомлення до сервісної шини зі встановленим часом активізації.

Схема архітектури розроблюваної системи є наступною:

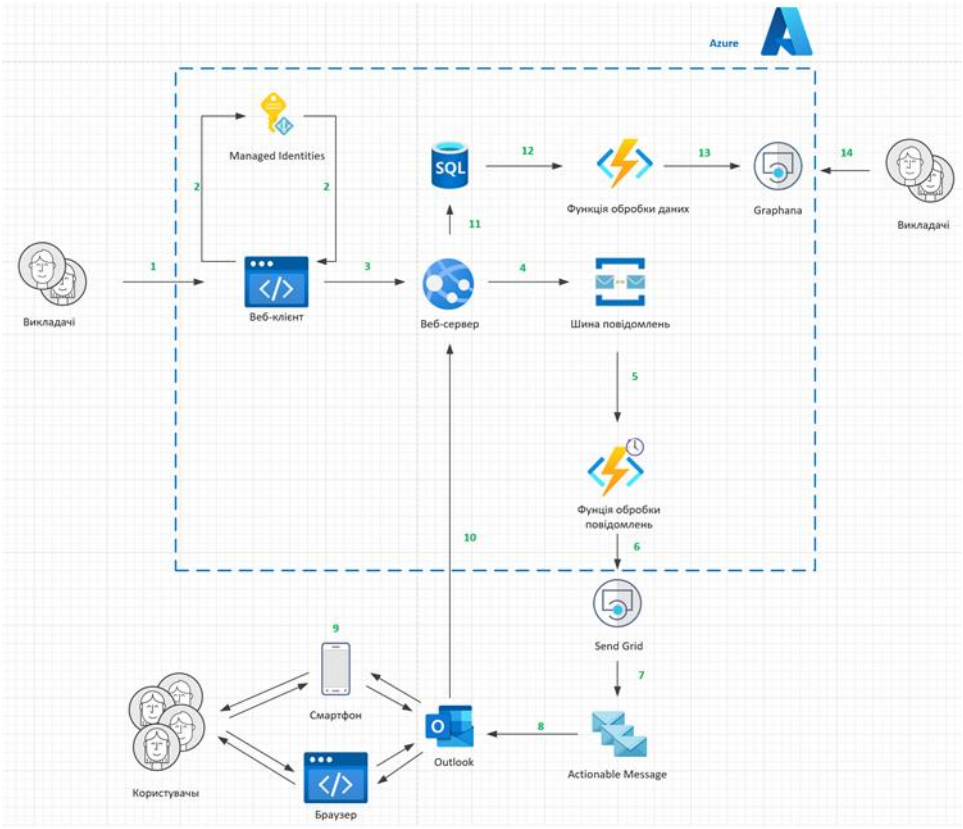


Рисунок.3.1 - Схема архітектури

Покроковий опис роботи архітектури: 1. Викладачі заходять на сервіс; 2. Використовуючи логін та пароль вони заходять в систему через Microsoft Identity; 3. Налаштовуючи необхідну форму відправляють дані на серверну частину; 4. Відправляється повідомлення в шину повідомлень з часом його активізації; 5 - 8. При активізації повідомлення функція-сервіс прочитає дане повідомлення і відправить на сервіс SendGrid, який відправить повідомлення на поштові адреси; 9 - 11. Отримавши відповіді від студентів, дані будуть відправлені на серверну частину і збережені до бази даних; 12 – 13. Дані зчитуються функцією обробки даних і відправлені на сервіс, який дасть можливість відобразити дані в зручному форматі.

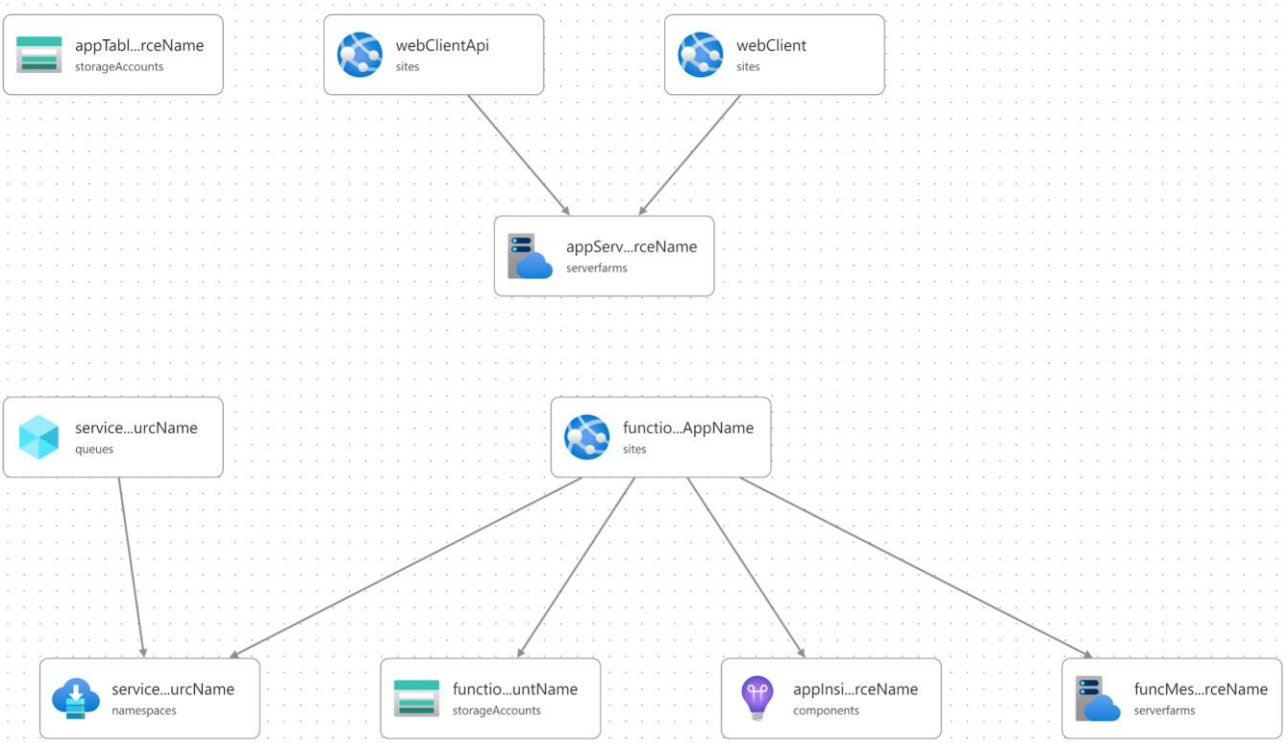


Рисунок 3.2 – Мапа використовуваних ресурсів Azure cloud

На даному зображені представленні використовувані у роботі ресурси Azure cloud, а також їх зв’язки. Кожен ресурс виконує свою функцію описану в покроковій роботі архітектури.

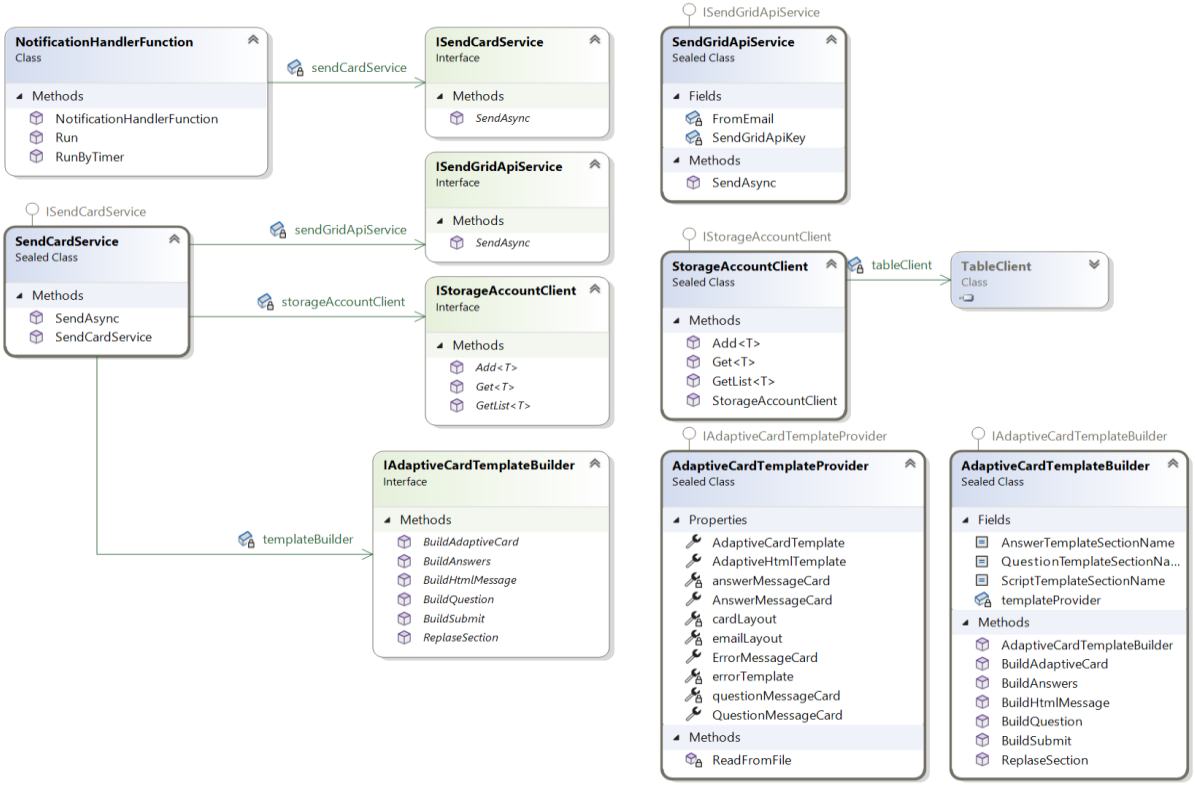


Рисунок 3.3 – Зображення структури класів та зв’язків у сервісах, що відповідають за створення та управління адаптивними картками

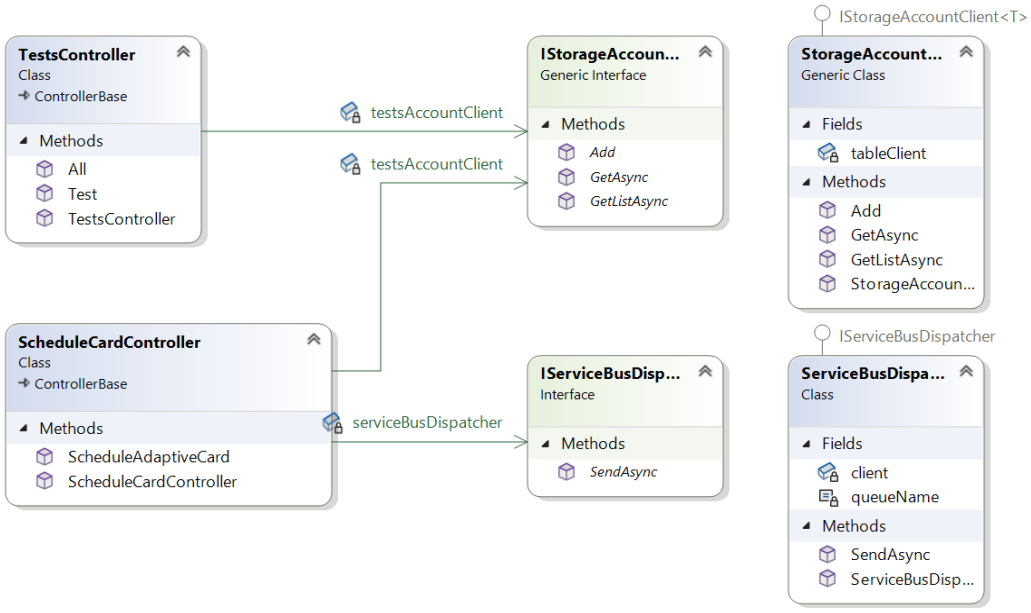


Рисунок 3.4 – Зображення структури класів та зв’язків у сервісах, що відповідають за обробку та управління адаптивними картками для веб сервісу

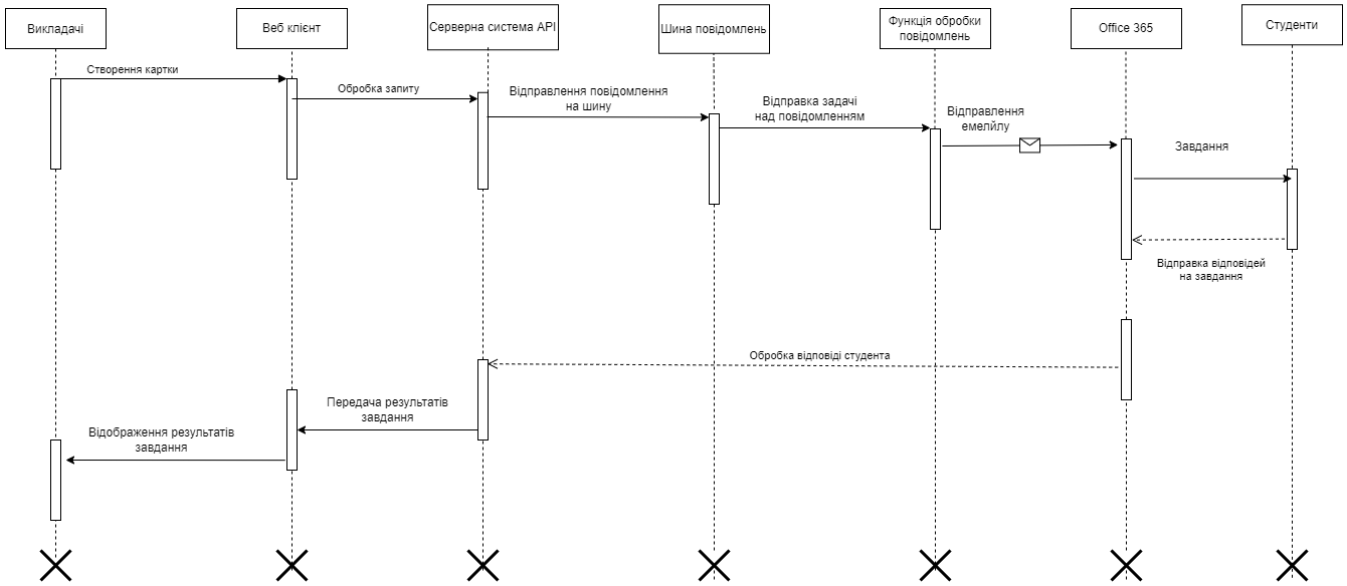


Рисунок 3.5 – Зображення діаграми послідовності роботи системи

Дана імплементація спрямована на те щоб полегшити, організувати і автоматизувати навчальний процес, такий як контроль, опитування та тестування.

При активізації повідомлення в шині, сервіс обробляє вхідний потік даних та відправляє форматований електронний лист через сервіс SendGrid на вказані поштові повідомлення. Кінцеві користувачі отримавши повідомлення мають змогу відправити форму, яка буде в середині, назад на сервер. Сервер обрахує дані і запише в базу даних до Storage Account.

Для відображення статистики створена функція (веб сервіс), яка перетворює дані в зручний формат для подальшого відправлення їх на сервіс, який зможе відобразити інформацію викладачеві в зручному форматі. Перейшовши до порталу буде відображена інформація зі змогою подальшого налаштування відображення за допомогою особливостей сервісу App Insights.

3.1 Побудова адаптивної картки

Адаптивна картка представляє собою розмітку описану за допомогою JSON нотації, якими служби і додатки можуть відкрито обмінюватися. Після доставки повідомлення, яке міститиме адаптивну картку, система, що приймає, перетворює розмітку на рідну структуру відображення, підпорядковуючись стандартам написання карток. Це допомагає розробити та інтегрувати зручний та легкий інтерфейс, який буде показаний користувачам на платформах, які використовуються та підтримують дану технологію.

Повідомлення на побудову і відправку електронного листа студенту отримується і перетворюється в об’єкт мови C# (рисунок 3.6) і передається в сервіс «SendCardService», щоб побудувати правильний форма картки.

```
5 references
public sealed class SendEmailMessage
{
    [JsonProperty("sendToEmail")]
    2 references
    public string SendToEmail { get; set; }

    [JsonProperty("teacherId")]
    1 reference
    public string TeacherId { get; set; }

    [JsonProperty("subjectName")]
    1 reference
    public string SubjectName { get; set; }

    [JsonProperty("questions")]
    1 reference
    public IEnumerable<SendEmailMessageQuestions> Questions { get; set; }
}
```

Рисунок 3.6 – Вхідний об’єкт для побудови картки

Об’єкт містить інформацію про те, кому відправляти повідомлення, ідентифікатор вчителя, назву предмету та список питань тесту, які були створені заздалегідь.

Сервіс «SendCardService» приймає об’єкт та працює з реалізацією інтерфейсу «IAdaptiveCardTemplateBuilder», щоб побудувати компоненти та сумістити їх в цільну адаптивну картку (рисунок 3.7).

```
33      public void Send(SendEmailMessage message)
34      {
35          int questionPossition = 0;
36
37          var questionsSections = message.Questions.Select(question =>
38          {
39              string questionString = this.templateBuilder.BuildQuestion(questionPossition++, question).EscapeRootArray();
40              var answersStrings = this.templateBuilder.BuildAnswers(question.Answers).Select(x => x.EscapeRootArray());
41
42              string questionTemplate = this.templateBuilder.ReplaseSection(
43                  questionString, string.Join(",", answersStrings),
44                  AdaptiveCardTemplateBuilder.AnswerTemplateSectionName);
45
46              return questionTemplate.EscapeRootArray();
47          });
48
49          string adaptiveCard = this.templateBuilder.BuildAdaptiveCard(string.Join(",", questionsSections), "", "", "");
50          string htmlTemplate = this.templateBuilder.BuildHtmlMessage(adaptiveCard);
51
52          var adaptiveCardState = new AdaptiveCardState
53          {
54              Template = adaptiveCard,
55              CardId = Guid.NewGuid().ToString(),
56              TeachedId = message.TeacherId,
57              StudentId = message.SendToEmail
58          };
59
60          this.storageAccountClient.Add(adaptiveCardState);
61          this.sendGridApiService.Send(htmlTemplate, message.SendToEmail, message.SubjectName);
62      }
```

Рисунок 3.7 – Метод побудови картки

Реалізація інтерфейсу «IAdaptiveCardTemplateBuilder» знаходить зарезервовані стрічки в описі адаптивної картки і заміняє їх на інформацію щодо тесту (рисунок 3.8).

```
2 references
32      public string BuildQuestion(int questionNumber, SendEmailMessageQuestions question) =>
33          this.templateProvider.QuestionMessageCard
34              .Replace("{{questionNumber}}", questionNumber.ToString())
35              .Replace("{{question}}", question.Question)
36              .Replace($@"{{{{""questionType""}}}}", question.Type == QuesntionType.Single ? "false" : "true");
37
```

Рисунок 3.8 – Метод побудови компоненту питань

Отримавши правильно форматовану адаптивну картку з даними тесту, інформація про картку зберігається в сховище для подальшого використання на серверному сервісі (рисунок 3.9).

```
49          string adaptiveCard = this.templateBuilder.BuildAdaptiveCard(string.Join(",", questionsSections), "", "", "");
50          string htmlTemplate = this.templateBuilder.BuildHtmlMessage(adaptiveCard);
51
52          var adaptiveCardState = new AdaptiveCardState
53          {
54              Template = adaptiveCard,
55              CardId = Guid.NewGuid().ToString(),
56              TeachedId = message.TeacherId,
57              StudentId = message.SendToEmail
58          };
59
60          this.storageAccountClient.Add(adaptiveCardState);
61          this.sendGridApiService.Send(htmlTemplate, message.SendToEmail, message.SubjectName);
```

Рисунок 3.9 – Збереження адаптивної картки до сховища

На даному етапі всі кроки виконанні і електронне повідомлення готове до відправи студентам.

Щоб забрати можливість повторного проходження тесту використовується вбудована особливість адаптивних карток, що дозволяє відправляти повідомлення на серверну частину при відкритті карти. Це дає можливість серверу визначити, чи користувач вже проходив даний тест та блокувати картку з повідомленням, що тест вже пройдений (рисунок 3.10).

```
77  "autoInvokeAction": {
78    "method": "POST",
79    "url": "{{refreshUrl}}",
80    "body": "",
81    "type": "Action.Http"
82  },
83  "version": "1.0",
84  "padding": "None"
85  }
```

Рисунок 3.10 – Метод автоматичного виклику серверної частини

Розмітка «autoInvokeAction» приймає параметри та використовує їх, щоб виконати виклик до серверної частини по адресу «url»[57].

3.2 Відправка адаптивної картки

Для відправки повідомлень використовується реалізація сервісу «ISendGridApiService», яка містить метод «Send» та приймає інформацію про отримувача, назва предмету, та тіло адаптивнох картки (рисунок 3.11).

```
17      public async Task Send(string messageBody, string recieverEmail, string subjectName)
18      {
19          var client = new SendGridClient(SendGridApiKey);
20
21          var fromEmail = new EmailAddress(FromEmail, "Notification App");
22          var toEmail = new EmailAddress(recieverEmail, "Student");
23
24          var subject = "Vntu Test";
25          var htmlContent = messageBody;
26
27          var msg = MailHelper.CreateSingleEmail(fromEmail, toEmail, subject, string.Empty, htmlContent);
28          var _ = await client.SendEmailAsync(msg).ConfigureAwait(false);
29      }
```

Рисунок 3.11 – Збереження адаптивної картки до сховища

Сервіс «Send» працює з клієнтом відправки повідомлень «SendGridClient» та відправляє повідомлення з визначеним заголовком, тілом повідомлення, та отримувачем. Адаптивна картка буде знаходиться в тілі повідомлення.

3.3 Реєстрація адаптивної картки

Технологія адаптивних карток має низку вимог, щоб дати дозвіл на використання та відображення їх в різних середовищах. Для відображення адаптивеної картки в середовищі Office 365 була проведена реєстрація в порталі розробників «Actionable Email Developer Dashboard» (рисунок 3.12), де вказується адерса відправника, шляхи серверної частини, що викликаються з інтерфейсу побудованої картки.

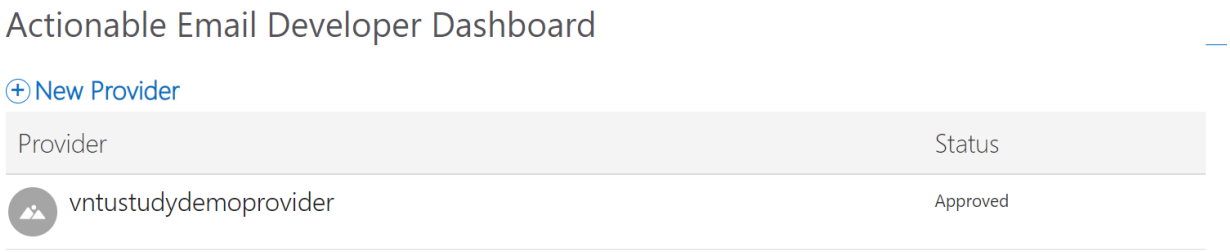


Рисунок 3.12 – Портал «Actionable Email Developer Dashboard»

При створені провайдера адаптивних карток відбувається процес верифікації, де член організації, що має адміністративні права, переглядає інформацію реєстрація та дає дозвіл. При отримані дозволу даний провайдер готовий до використання.

3.4 Використання Azure Function

В ході даної роботи картки будуються відповідно даним, які отримуються на сервіс, що відповідає за побудови карток. Сервіс нотифікації представляє з себе Azure Function з інтерфейсом читання повідомлень з шини повідомлень (рисунок 3.13).

```
[FunctionName("notification-handle")]
0 references
public void Run([ServiceBusTrigger(
    "send-message-scheduled-queue",
    Connection = "notification-service-bus-connection-string")] string queueItem, ILogger log)
{
    SendEmailMessage queueMessage = JsonConvert.DeserializeObject<SendEmailMessage>(queueItem);

    this.sendCardService.Send(queueMessage);

    log.LogInformation($"C# ServiceBus queue trigger function processed message: {queueItem}");
}
```

Рисунок 3.13 – Зображення коду Azure Function

Метод функції приймає параметри, які описують джерело отримання повідомлень та стрічка, яка міститиме тіло повідомлення. Для подальшої роботи

стрічка перетворюється на об’єкт мови програмування, який зрозумілий сервісами коду.

Останнім кроком є логування події читання повідомлення. Це дає можливість відслідковувати кількість оброблених повідомлень та дає можливість використання цього значення при побудові комплексних аналітичних даних.

3.5 Серверна частина додатка – Web API

Одним із компонентів мікросервісної системи додатку є серверна частина, яка незалежно розгорнута в середовищі Azure. Ціль серверного компоненту є обробка повідомлень з клієнтської частини та з частини адаптивних карток, обробка інформації та збереження цієї інформації в сховище для подальшого використання.

В основі веб сервісу лежить технологія Asp DotNet 6 та всі суміжні бібліотеки та технології, що дають можливість побудувати кінцеві точки серверної частини без надлишкового коду та в зручній манері.

Одним з головних компонентів побудови додатку є використання Inversion Of Control [58] контейнерів, що забирають зв’язність компонентів між собою і розділяють абстракцію від реалізації. Це надає розробнику можливість змінювати реалізацію компоненту в одному місці і не витрачати час на пошуки місць використання об’єкту.

З використанням Inversion Of Control контейнерів відбувається реєстрація компоненту, що дає змогу використовувати цей компонент в будь яких місцях, передаючи абстракцію в конструктор іншого компоненту(рисунок 3.14).

```
[Route("api/[controller]")]
[ApiController]
1 reference
public class CardsController : ControllerBase
{
    private readonly IStorageAccountClient<AdaptiveCardState> cardStateStorageClient;

    0 references | 0 exceptions, - live
    public CardsController(CardStateStorageClient cardStateStorageClient)
    {
        this.cardStateStorageClient = cardStateStorageClient;
    }

    [HttpGet("{cardId}")]
    0 references | 0 requests, - live | 0 exceptions, - live
    public async Task<IEnumerable<CardStateResponse>> All([FromRoute] string cardId)
    {
        var cardsStates = await this.cardStateStorageClient.GetListAsync(cardId);

        return cardsStates.Select(x => new CardStateResponse
        {
            Score = x.Score,
            StudentFio = x.StudentFio,
            UserEmail = x.UserEmail
        });
    }
}
```

Рисунок 3.14 – Використання «CardStateStorageClient»

Контроллер «CardsController» приймає об’єкт класу «CardStateStorageClient», який унаслідується від абстракції «IStoageAccountClient». Цим самим не відбувається створення об’єкту напряму у коді, натомість ініціалізація відбувається за допомогою IoC контейнерів та передається в конструктор контролера. Це дає змогу забрати звязність компонентів та працювати з абстракція, а не з реалізаціями компонентів, що є еталонним підходом в розробці сучасних додатків.

Для збереження інформації нового тесту створений клас «FormsController», який представляє собою клас, що унаслідується від «ControllerBase», який надає класу можливості написання методів, які інтерпретуються як кінцеві точки HTTP. Клас там метод контролеру містить атрибути, які описують шлях кінцевої точки HTTP (Рисунок 3.15).

```
[Route("api/[controller]")]
[ApiController]
1 reference
public class TestsController : ControllerBase
{
    private readonly IStorageAccountClient<TestsInformation> testsAccountClient;

    0 references | 0 exceptions, - live
    public TestsController(TestsStorageClient testsAccountClient)...

    [HttpGet("{teacherId}")]
    0 references | 0 requests, - live | 0 exceptions, - live
    public Task<IEnumerable<TestsInformation>> All([FromRoute] string teacherId)...

    [HttpPost()]
    0 references | 0 requests, - live | 0 exceptions, - live
    public ActionResult Test([FromBody] NewTestRequest testRequest)...
}
```

Рисунок 3.15 – Контроллер тесту

Контролер містить два методи, необхідні для роботи клієнтського додатку:

- Метод отримання тестів вчителя по ідентифікатору вчителя
- Метод зберігання тесту в сховище додатку

Також даний компонент системи містить контролер відправки повідомлення у визначений час та контролер який дає змогу отримати інформацію про вже існуючи адаптивні картки, що відправлені користувачам.

3.6 Інфраструктура додатку

Для створення інфраструктури додатку використовується «bicep», та «yaml» розмітка, яка описує кроки виконання під час процесу «Continuous Integration & Continuous Deployment».

Створено окрему категорію в структурі проекту, яка зберігає інформацію про необхідні файли та виконавчі скрипти. (Рисунок 3.16)

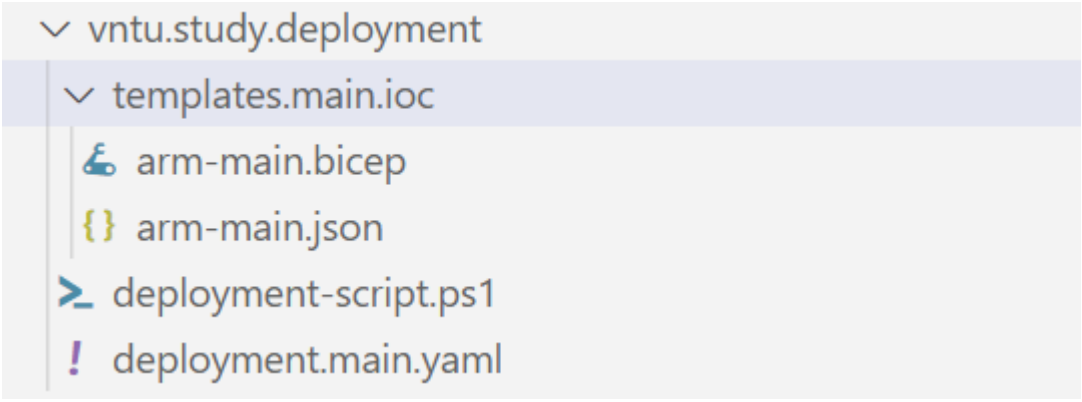


Рисунок 3.16– Каталог інфраструктури

В даному каталозі файлів розміщено наступні файли:

- deployment.main.yaml – описує кроки виконання дій в системі «Azure Devops»;
- deployment script.ps1 – скрип виклику «arm-main.bicep»;
- arm-main.bicep – розмітка, що містить інформацію про інфраструктуру додатку.

Головним файлом інфраструктури вважається є «arm-main.bicep», що містить інформацію, яка описує і розгортає наступні додатки в хмарному середовищі Azure (рисунок 3.17):

- ai-vntu-study-northeurope;
- as-vntu-study-api-northeurope;
- as-vntu-study-client-northeurope;
- asp-as-vntu-study-api-northeurope;
- asp-fn-vntu-study-msg-handle-northeurope;
- fn-vntu-study-msg-handle-northeurope;
- savntuappstorage;
- savntumsghandlefnc;
- sb-vntu-study-northeurope.

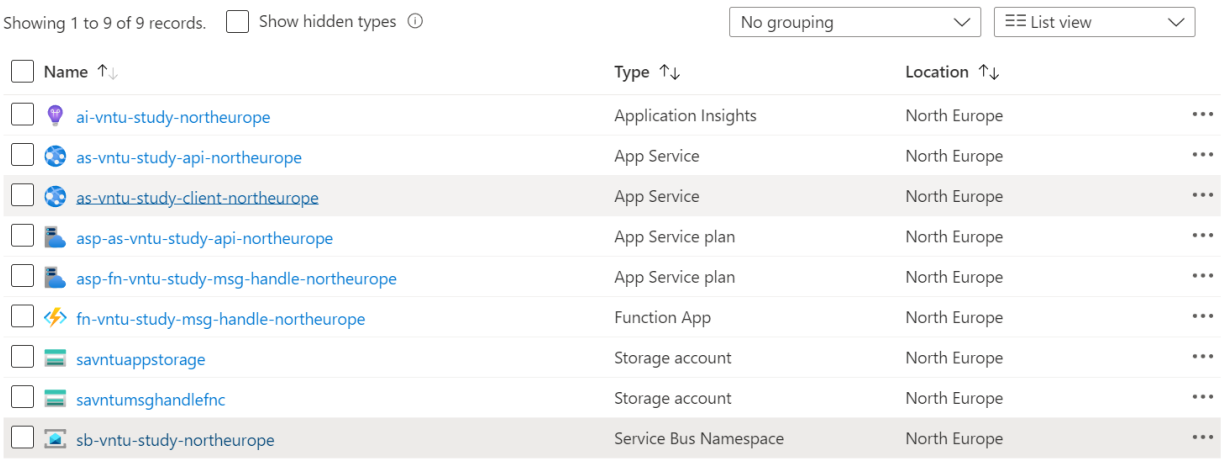


Рисунок 3.17 – Розгорнуті сервіси в хмраному середовищі «Azure»

Кожний сервіс описується декларативним методом в файлі «bicep», описуючи бажаний результат, а не метод побудови розгортання сервісу.

Опис кожного ресурсу має однакову загальну структуру, де описується вся необхідна інформація про ресурс, що потрібно створити:

```
resource webClientApi 'Microsoft.Web/sites@2021-02-01' = {
  name: webClientApiName
  location: resourceLocation
  properties: {
    httpsOnly: true
    serverFarmId: appServicePlanPortalResourceName.id
    siteConfig: {
      minTlsVersion: '1.2'
      ftpsState: 'FtpsOnly'
    }
  }
  identity: {
    type: 'SystemAssigned'
  }
}
```

Код 3.1 – Опис ресурсу «as-vntu-study-api-northeurope»

В даному коді описується регіон розташування сервісу, особливості, ідентифікатор ресурсу плану та ім’я. Параметри, що розташовані в категорії «properties» відрізняються і описують властивості притаманні описуючому ресурсу.

```
resource serviceBusQueueResourcName 'Microsoft.ServiceBus/Namespaces/Queues@2021-11-01' = {
  parent: serviceBusNamespaceResourcName
  name: 'send-message-scheduled-queue'
  properties: {
    lockDuration: 'PT5M'
    maxSizeInMegabytes: 1024
    requiresDuplicateDetection: false
    requiresSession: false
    defaultMessageTimeToLive: 'P10675199DT2H48M5.4775807S'
    deadLetteringOnMessageExpiration: false
    duplicateDetectionHistoryTimeWindow: 'PT10M'
    maxDeliveryCount: 10
    autoDeleteOnIdle: 'P10675199DT2H48M5.4775807S'
    enablePartitioning: false
    enableExpress: false
  }
}
```

Код 3.2 – Опис ресурсу «sb-vntu-study-northeurope»

Категорія «properties» містить параметри, що описують роботу сервісу шини повідомлень. Параметри визначають на скільки часу буде блокуватися повідомлення в межах одного читання, величина одного повідомлення, використання особливостей сервісу як сесійність, виявлення дуплікацій тощо.

Для автоматизації розгортання сервісів та доставки коду до цих ресурсів використовується сервіс Azure Devops, який читає файл «yaml» з описом кроків, що виконуються під час додавання змін в головну вітку система контроля версій.

Кожний крок описується за допомогою «yaml» розмітки.

YAML – це мова серіалізації даних, яка часто використовується для написання конфігураційних файлів.

Для написання завдання побудови вихідних файлів додатку, використовується «DotNetCoreCLI@2» завдання «Azure Devops», що викликається та описується за допомогою «yaml»:

```
        - task: DotNetCoreCLI@2
      displayName: "dotnet publish"
      inputs:
        command: "publish"
        publishWebProjects: false
        projects: |
```

```
 **/*vntu.study.webapi/*.csproj
 **/*vntu.study.notification/*.csproj
 **/*vntu.study.webapp/*.csproj
arguments: "--configuration $(BuildConfiguration) --output"
zipAfterPublish: true
modifyOutputPath: true
```

Код 3.3 – Крок збирання вихідних файлів додатку

Слідуючи даним правилам написання кожного пункту виконання, створений файл «deployment.main.yaml», що описує послідовність і взаємодію кроків.

Процес запуску кроків відбувається при отримані змін в головну вітку, після чого створюється подія, що повідомляє систему розгортання про надходження змін. Система розгортання реагує на зміни, читає правила та кроки описані в файлі «yaml» та створює ресурси в хмарному середовищі «Azure». Виконавши попередні кроки, програмний код компілюється з формуванням вихідним файлів та вивантажуються до створених ресурсів (рисунок 3.18).

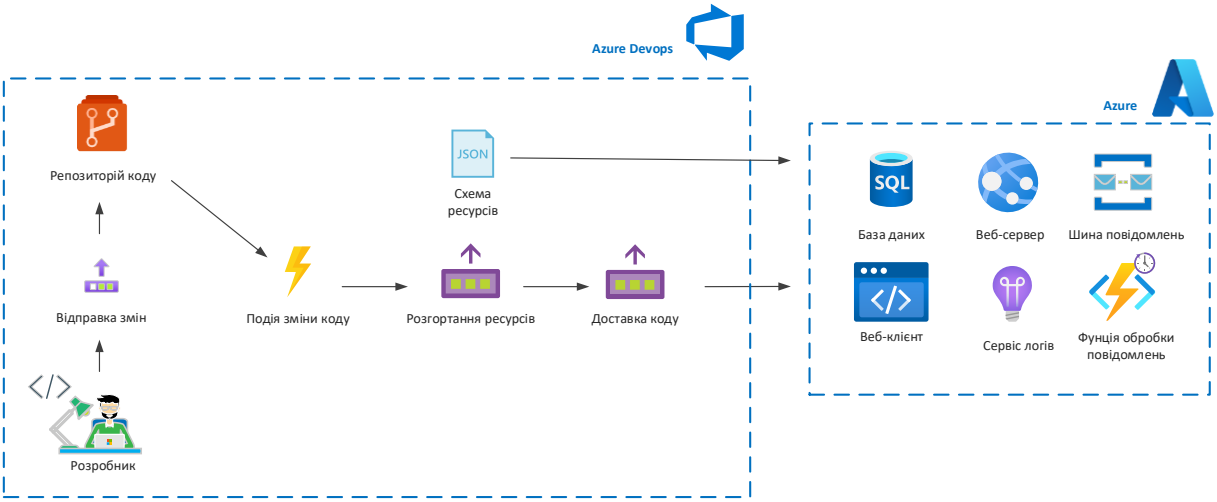
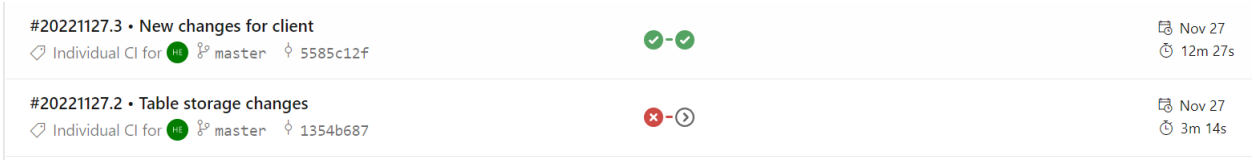


Рисунок 3.18 – Схема розгортання ресурсів та доставка коду

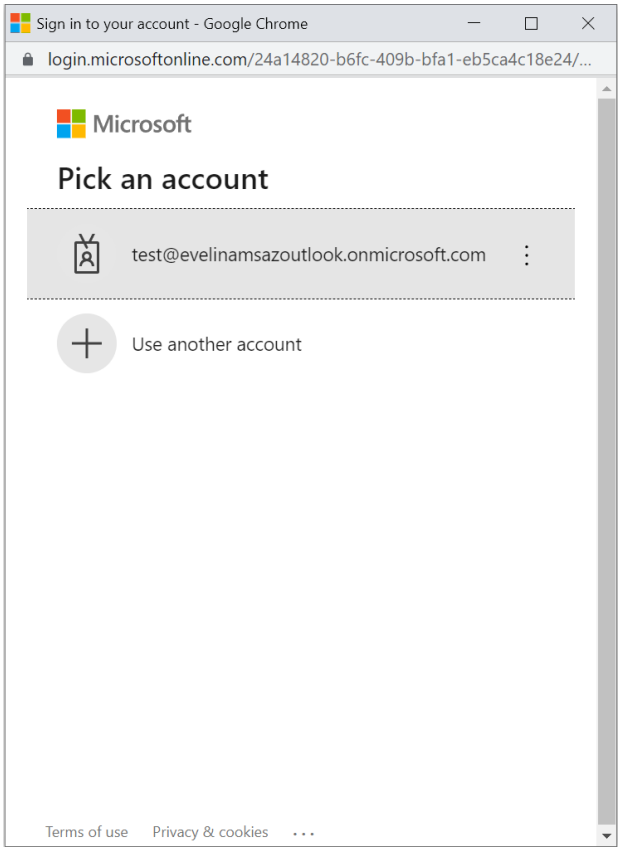
Кроки розгортання інфраструктури та доставки коду кожних нових змін можна переглянути в порталі «AzureDevops» і при отриманні помилки отримати розгорнуту інформацію про причини виникнення. (рисунок 3.19)



Зображення 3.19 – Хід виконання розгортання інфраструктури і доставка коду

3.7 Система авторизація викладача

Для авторизації вчителів використовується вбудована система авторизації і автентифікації Azure, яка має назву Azure Active Directory від Microsoft Identity. Дана система дозвоялє налаштувати метод входу та обмежити доступ до певного ресурсу. В розробці адаптивних карт використовуються налаштування, де війти в систему мають змогу лише користувачі цього ж Azure Active Directory. Перейшовши на клієнтську частину додатку, користувач отримає вікно логіну і паролю, щоб ввести свої дані (рисунок 3.20).



Зображення 3.20 – Вікно логіну користувача до порталу додатку

Ввівши дані, користувач буде перенаправлений на портал додатку з можливість переглядати захищені дані.

Для використання можливості авторизації користувачів за допомогою Azure Active Directory створений Azure Active Directory додаток в якому вказується що за додаток буде використовувати систему авторизації та які права потрібні за допомогою токену, що генерується після логіну.

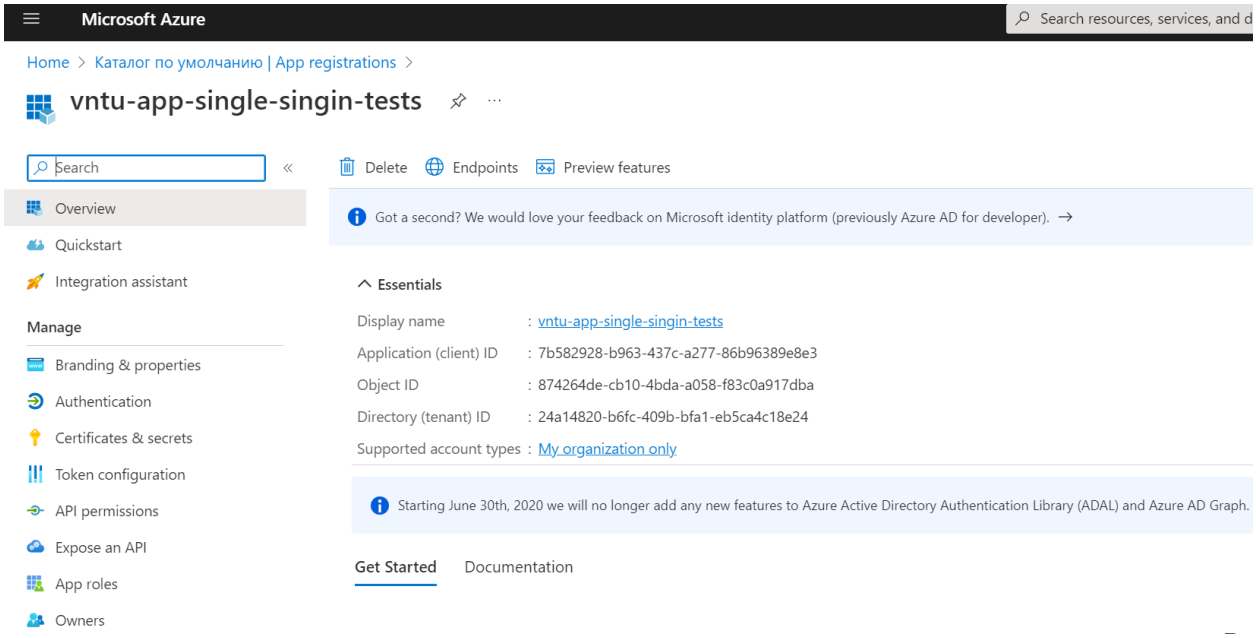


Рисунок 3.21– Вікно інформації Azure Active Directory додатоку

Інформація яка відображається в Azure Active Directory додатоку переноситься в код та використовується при логуванні. (рисунок 3.22)

Рисунок 3.22– Інформація Azure Active Directory додатку в коді

3.8 Розробка клієнтської частини

Клієнтська частина додатку написана з використанням нової технології Blazor, розроблена Microsoft. Використання даної технології дає можливість писати сучасні javascript додатки, не використовуючи javascript. Володіючи мовою C# і розміткою HMTL, створюються додатки, що транслюються в javascript.

Структура клієнтського додатку складається файлів мови програмування C# та файлів HTML, які розділені структурно по каталогам. (рисунок 3.23)

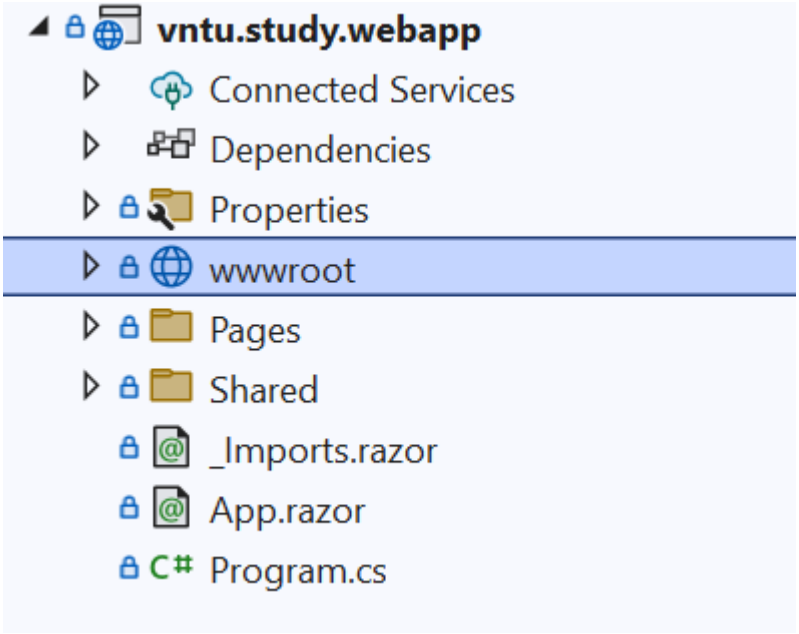


Рисунок 3.23 – Структура клієнтського додатку

Для реалізації всіх можливостей додатку та можливості перегляду результатів створено наступні сторінки сторінки:

- TestCreation.razor
- TestResults.razor
- TestSchedule.razor

«TestCreation» сторінка має інтерфейс створення тесту та введення всіх необхідних даних в систему.

«TestResuls» сторінка відображає наявні результати по кожному тесту викладача.

«TestSchedule.razor» сторінка яка дає можливість відправити вибране повідомлення студентам в зазначений час.

Кожна сторінка має свій ідентифікатор сторінки, по якому сторінка доступна: «@page "/"». Також кожна секція містить секцію «@code», яка містить додаткову логіку на мові програмування C#, цим самим замінюючи javascript (рисунок 3.24).

```
1   @page "/counter"
2
3   <PageTitle>Counter</PageTitle>
4
5   <h1>Counter</h1>
6
7   <p role="status">Current count: @currentCount</p>
8
9   <button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
10
11  @code {
12      private int currentCount = 0;
13
14      private void IncrementCount()
15      {
16          currentCount++;
17      }
18  }
19
```

Рисунок 3.24 – Приклад структури сторінки Blazor

Комбінуючи сторінки додатку, був створений портал вчителя, що дає можливість створити, відправити, та переглянути результати адаптивних карток.

3.9 Тестування програмного продукту

У ході розробки будь якого продукту, замовники та розробники часто зіштовхуються з різними проблемами і причинами помилок, які б могли бути виявлені на ранніх стадіях програми. Все це показує важливість такого процесу як «тестування програмного забезпечення». Є очевидним, що чим пізніше почате тестування програмної системи, тим є вищі ризики, тим менш надійною вона може вийти.

Тестування програмного забезпечення є технологічним процесом перевірки системи, продукту на наявність помилок чи не відповідність до зазначених вимог та критеріїв.

Типів тестування є безліч, вони суттєво різняться за метою та бажаним результатом по закінченню. Відрізняють за завданням, які з їх допомогою вирішуються, і по техніці, що використовується. Їх можна класифікувати у відповідності до стандартних показників якості, які перевіряються за їх допомогою. При перевірці функціональності програмного забезпечення використовуються власне функціональні тести, а також тести безпеки, обсягу та інші.

Вимогами до даної розробки є: зручний графічний інтерфейс сайту викладача, робоче меню вибору основних функцій, можливість створення та відправки інтерактивних форм, отримання актуальної форми із коректним змістом та дизайном, змога надати відповідь.

Першим кроком викладач авторизується та заходить на веб-сайт, портал викладача. Авторизація відбувається тоді, коли у викладача є створений акаунт у Microsoft та доданий до ресурсів системи в Azure portal. Портал викладача доступний за посиланням: https://as-vntu-study-client-northeurope.azurewebsites.net/ .

Сайт має зручний та простий для користувача інтерфейс. Присутні 3 сторінки: «домашня», створення тесту та відправлення тесту. Логотип чи той самий «Favicon» відповідає розроблюваному сайту (рисунок 3.26).

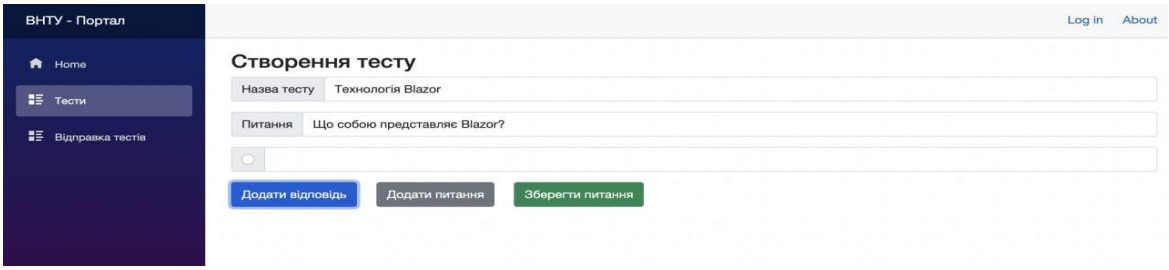


Рисунок 3.25 – Зображення інтерфейсу порталу викладача: сторінка створення тесту

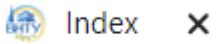


Рисунок 3.26 - Зображення «Favicon» сайту

Створення тесту є досить легким і робиться у декілька кліків:

1. Необхідно створити назву необхідного тесту (рисунок 3.25).
2. Натиснути «Додати питання», що створити питання (рисунок 3.25).
3. Щоб додати варіанти відповідей, необхідно вибрати «Додати відповідь» (рисунок 2.27).
4. Вибрати правильну відповідь можна за допомогою «тогла». Натисніть на кружок відповіді, що є правильна (рисунок 2.27).
5. Після закінчення додавання відповідей можна натиснути «Додати питання» для наступного питання чи «Зберегти питання», щоб зберегти його і продовжити далі (рисунок 2.28).

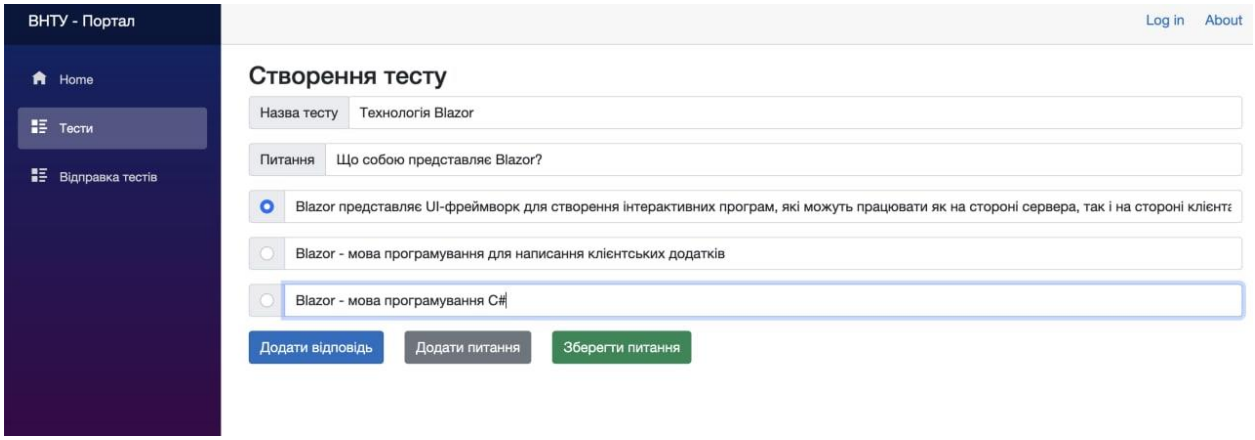


Рисунок 3.27 – Зображення прикладу створення тесту

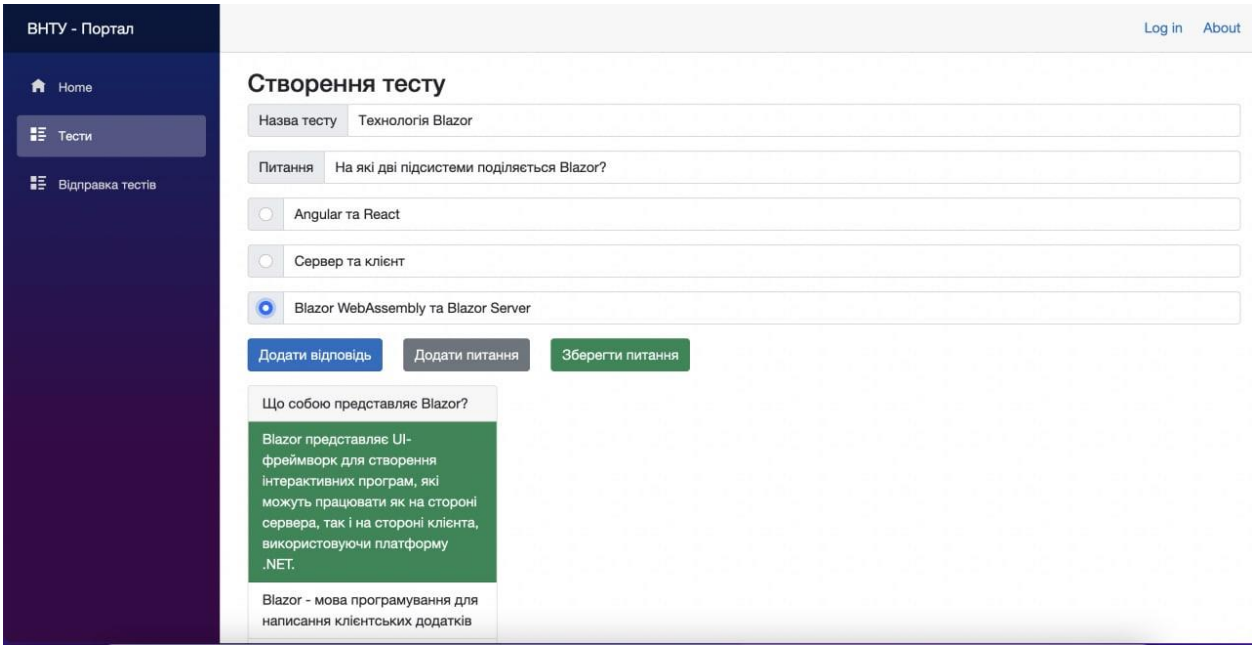


Рисунок 2.28 – Зображення прикладу роботи сайту при створенні тесту

Після додавання тестової картки, вона повинна автоматично відобразиться у переліку готових тестів до відправки на наступній сторінці.

Щоб відправити необхідний тест для студентів, варто перейти до наступної сторінки «Відправка тестів». Оберіть необхідний тест із доданих у списку. Зазначте пошти студентів, кому потрібно відправити тест. Відправите тест (рисунок 3.30).

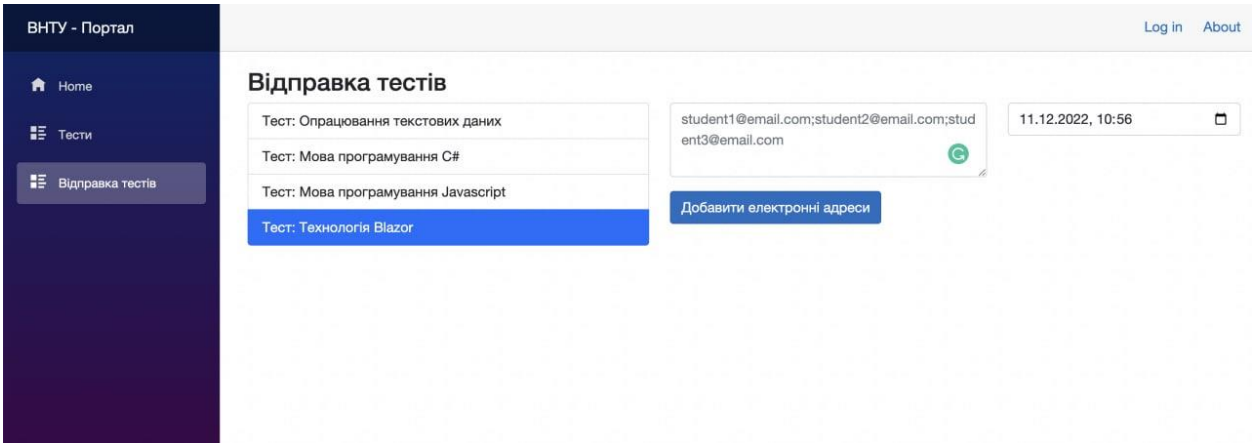


Рисунок 3.30 – Зображення сторінки «відправки тестів»

Після відправки, на сторінці відобразиться строка із часом та датою відправки тесту, відобразиться кількість відповідей но нього. Дана строка буде оновлюватись в реальному часі, так як будуть приходити відповіді від студентів (рисунок 3.31).

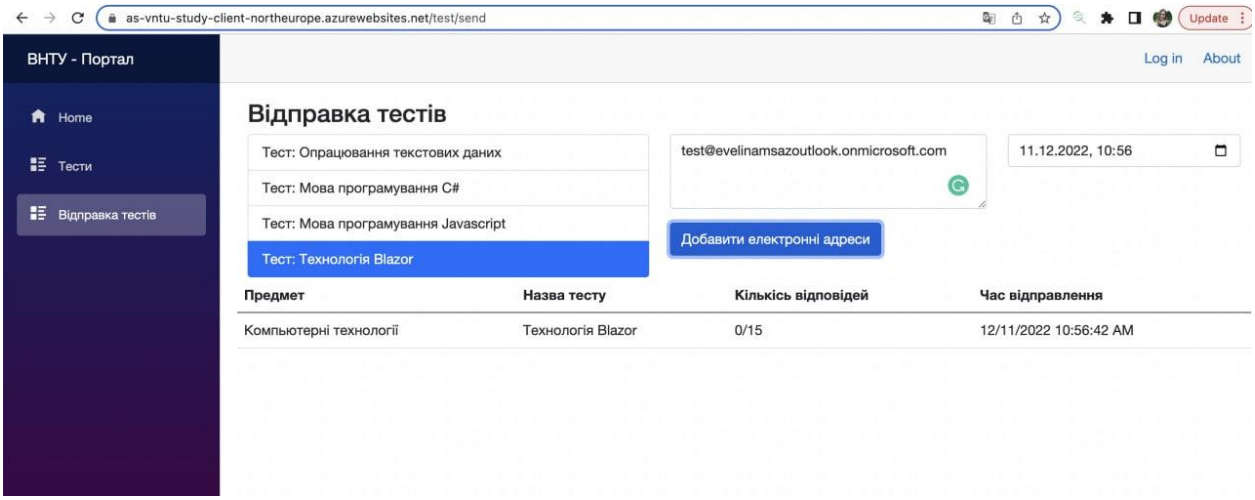


Рисунок 3.31 – Зображення відправленого тесту та строки його стану.

Результатом відправки є отримана форма на пошту студента у Outlook від Microsoft Office 365. Студент може надати відповідь на форму у зручний спосіб з телефону чи ком’ютеру, у певний визначений викладачем час.

Інтерактивна форма є зручна і адаптована під будь-який девайс.

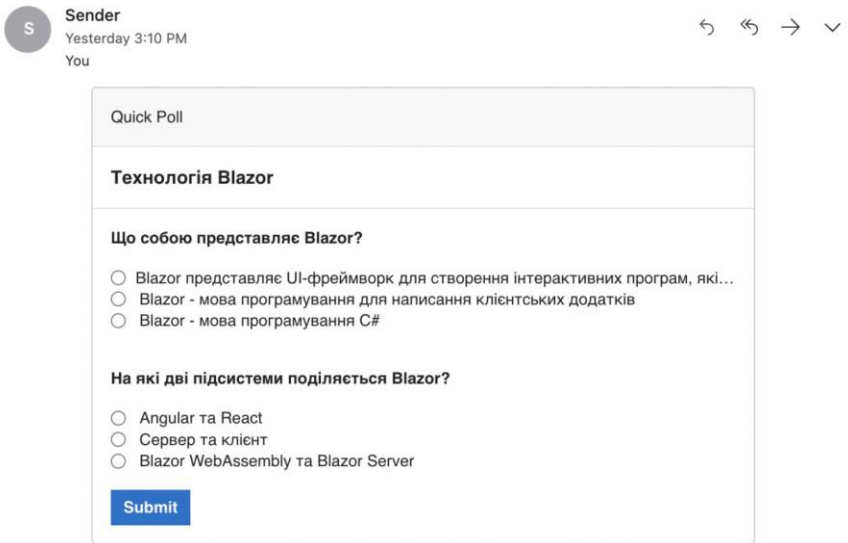


Рисунок 3.32 – Скрін інтерактивної форми із комп’ютера

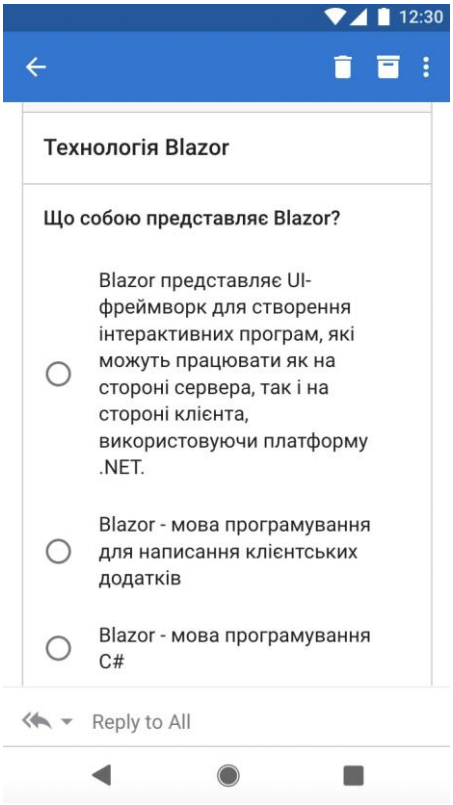


Рисунок 3.33 – Скрін інтерактивної форми із телефону

Також форму можна редагувати в дизайнері Actionable Message Designer, налаштовувати вигляд за бажанням. Налаштування доступне через дизайнер із зручними інструментами, а також через файловий редактор у json форматі.

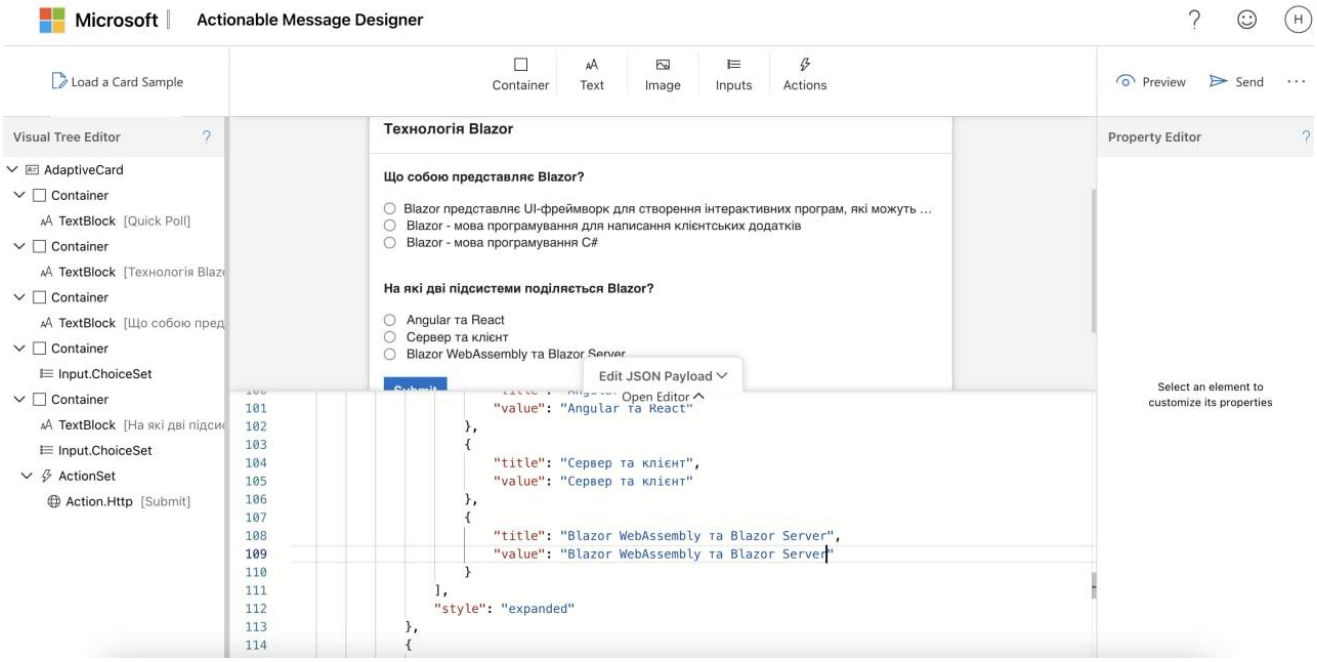


Рисунок 3.34 – Зображення форми із дизайнера Actionable Message Designer

Останнім етапом є аналітика даних відповідей студентів. Вона є доступною у Azure portal App Insights.

Дана імплементація спрямована на те щоб полегшити, організувати і автоматизувати навчальний процес, такий як контроль, опитування та тестування.

3.10 Висновки

Опираючись на головну задачу дослідження, було розроблено архітектуру додатку, для автоматизації навчального процесу: розробка інтерактивних електронних поштових повідомлень від Microsoft Azure. Описано логіку інтерактивних повідомлень: реєстрація та відправка, середовище їх розробки, які розроблялись у Microsoft Visual Studio 2022, мовою програмування: C#, технологією: .NET6, та у хмарному середовищі Azure. Для веб порталу використовується Microsoft Identity.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного продукту інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу. Ціллю розробки є створення автоматизованої системи тестування з метою оцінювання знань студентів з певного предмету. Особливістю програми є те, що дана технологія використовує адаптивні картки поштових повідомлень Office 365 з можливістю модернізації та оцінювання студентів в реальному часі. А налаштування і побудова електронної форми виконується у будь-якому форматі за правилами структури адаптивних карт, використовуючи хмарне середовище.

Аналогом даної розробки є Survey Monkey, ціна 48000 грн/рік; TestMasters, ціна 32000 грн. і витрати на спеціалізовану систему ще 29600 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції					

1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах

Продовження табл. 4.1

Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційни х витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційн і витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою

7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 4.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4.2

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	4	3	4
Наявність аналогів на ринку	4	4	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	3	3
Супровідна документація	3	4	4
Сума	44	42	43
Середньоарифметична сума балів	(44+42+43) / 3 = 43		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

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

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що програмний продукт відрізняється від існуючих тим, що дана технологія є автоматизованою системою тестування з ціллю оцінювання знань студентів з певного предмету. Особливістю програми є те, що дана технологія використовує адаптивні картки поштових повідомлень Office 365 з можливістю модернізації та оцінювання студентів в реальному часі. А налаштування і побудова електронної форми виконується у будь-якому форматі за правилами структури адаптивних карт, використовуючи хмарне середовище.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

4.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \quad , \tag{4.1}$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$Т_р$ – число робочих днів в місяці, 20 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.1.

Таблиця 4.1 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	34000	1700,00	40	68000,000
Програміст	30000	1500,00	40	60000,000
Всього				128000,00

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту

4.2.2 Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 15 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 15\ \% / 100\ \% \tag{4.2}$$

$З_д = (128000{,}00 \cdot 15\ \% / 100\ \%) = 19200{,}00$ (грн.)

4.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (4.3)$$

$Н_з = (128000{,}00 + 19200{,}00) \cdot 22\ \% / 100\ \% = 32384{,}00$ (грн.)

4.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

4.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \qquad (4.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп’ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 2,00

міс.

$$A_{обл} = \frac{20000}{2} \times \frac{2{,}00}{12} = 1666{,}67 \ грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.2. Для розрахунку амортизації нематеріальних ресурсів використовується формула:

$$А_{н.р.} = Ц_{Н.р.} * Н_а * \frac{t_{вик}}{12} \quad A_{н.р.} = Ц_{н.р.} * H_a * \frac{t_{вик.}}{12} \qquad (4.5)$$

Але, так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $B_{нем.ак.}$ = 2480 грн.

Таблиця 4.2 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Комп'ютер та комп'ютерна переферія	20000	2	2,00	1666,667
Офісне обладнення	22000	4	2,00	916,667
Приміщення	1000000	20	2,00	8333,333
Всього				10916,67

5.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить

від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.6)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,5 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$В_е = 0{,}9 \cdot 0{,}5 \cdot 8 \cdot 40 \cdot 6{,}2 = 892{,}8$ (грн.)

5.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.7)$$

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

$I_в$ = 128000,00 * 100% / 100% = 128000 (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$H_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \qquad (4.8)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$H_{нзв}$ = 128000,00 * 130 % / 100 % = 166400 (грн.)

5.2.9 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$B_{заг}$ = 128000,00+19200,00+32384,00+10916,67+2480+892,80+128000+
+166400 = 488273,47 грн.

5.2.11 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (5.9)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

ЗВ = 488273,47 / 0,5 = 976547 грн.

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб'єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

- абсолютного економічного ефекту (чистого дисконтованого доходу);
- внутрішньої економічної дохідності (внутрішньої норми дохідності);
- терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

4.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.10)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

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

$Цб$ – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

ρ – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 8000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *800* грн. Кількість одиниць реалізованої продукції

також збільшиться: протягом першого року – на 10000 шт., протягом другого року – на 5000 шт., протягом третього року на 3000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

*ΔП₁ = (0*800 + (8000 + 800)*10000)* 0,8333* 0,2) * (1 - 0,18) = 10933332,896 грн.*

*ΔП₂ = (0*800 + (8000 + 800)*(10000+5000)* 0,8333* 0,2) * (1 - 0,18) = 18039999,278 грн.*

*ΔП₃ = (0*800 + (8000 + 800)*(10000+5000+3000)* 0,8333* 0,2) * (1 - 0,18) = 21647999,134 грн.*

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 50621331,31 грн.

4.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= (10933332,896/(1+0,1)^1)+(18039999,278/(1+0,1)^2)+(21647999,134/
/(1+0,1)^3) = 9939393,54 + 14909090,31 + 16264462,16 = 41112946,01 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (4.12)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

PV = 2 * 976547 = 1953093,87 *грн.*

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \tag{4.13}$$

$E_{абс}$ = 41112946,01 – 1953093,87 = 39159852,15 грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розра-хувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

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

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

$T_ж$ – життєвий цикл наукової розробки, роки.

$E_в = \sqrt[3]{(1 + 39159852{,}15/1953093{,}87} - 1 = 1{,}761$

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

$$\tau = d + f, \tag{4.15}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f−показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

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

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

$$T_{ок} = \frac{1}{E_в}, \tag{4.16}$$

$T_{ок}$ = 1 / 1,761 = 0,57 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,57 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 976547 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,57 роки.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи було проведено аналіз усіх існуючих аналогів засобів автоматизації навчального процесу, а саме проведення оцінювання знань студента, визначено потреби користувачів. Було поставлено задачу дослідження. За результатами дослідження було розв’язано наукову задачу, щодо автоматизації навчального процесу.

Було проведено порівняння існуючих принципів та підходів розробки програмного забезпечення з використанням хмарних середовищ, методів автоматизації навчального процесу і їх ефективне використання. Визначено основні переваги та недоліки даної розробки.

Розглянуто методи функціонального, інформаційного моделювання, основ та принципів проектування програмних систем для забезпечення головної мети роботи, що включає в себе зменшення витрат часу та ресурсів на проведення тестування, покращення якості оцінювання студента та автоматизацію програмного продукту.

В ході виконання роботи було автоматизовано створення інтерактивних карток (опитувальника, завдання), їх відправка та обробка, створено веб-сайт для викладача та додано аналітичну систему результатів, в основі яких використовується Microsoft Adaptive cards, який базується на хмарному середовищі Azure та Microsoft Actionable Messages. Для збереження даних використано Azure Storage Account, що дозволяє зберігати дані безкоштовно до певного ліміту. Системою аналітики є Microsoft AppInsights. Клієнтська частина написана на Blazor, використано Azure Function, сервіс розсилання поштових електронних листів – Send Grid.

Використовуючи Azure Cloud & azure devops було створено автоматичне розгортання середовища, що дозволяє створити все з нуля, в випадку необхідності.

Написання коду виконано у Microsoft Visual Studio 2022, мова програмування C#, технологія .NET6.

Головним аспектом є те, що рішення задачі розробки використовує безкоштовні сервіси Azure Cloud та Azure Devops, що робить даний продукт не фінансово залежним.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Горбачова Е.О, Ковалюк О.О., Розробка інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу// ВНТУ. – 2022. – [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2022/paper/view/14206
2. L Науково-технічна конференція факультету комп'ютерних систем і автоматики (2021). ВНТУ. Ковалюк О.О., Горбачова Е.О.. Розробка чат-боту з використанням хмарних сервісів для автоматизації навчального процесу. [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2021/paper/view/12567/10483
3. Одеський Національний Економічний Університет. Пархуць Л., Ясинська С. ІНФОРМАЦІЙНІ СИСТЕМИ В ОСВІТІ: АВТОМАТИЗОВАНІ НАВЧАЛЬНІ СИСТЕМИ. [Електронний ресурс]. URL: https://core.ac.uk/download/pdf/147037367.pdf
4. В.І. Каркульовський, А.Б. Керницький1 , І.І. Мотика, Б.В. Каркульовський, Я.П. Кісь. Національний університет “Львівська політехніка“, кафедра систем автоматизованого проектування, кафедра інформаційних систем та мереж. АВТОМАТИЗОВАНА НАВЧАЛЬНА СИСТЕМА “ЧИСЛОВІ МЕТОДИ В ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЯХ”. [Електронний ресурс]. URL: http://vlp.com.ua/files/04_21.pdf
5. ГОРБАТЮК Р. Тестування як метод педагогічного контролю якості знань майбутніх фахівців із вищою освітою / Р. ГОРБАТЮК, Т. СІТКАР // Нова педагогічна думка. — 2014. — Т. 3, No. 3. — Ст. 179–183.
6. Федорова А. І. Тестова перевірка знань студентів : плюси та мінуси (на прикладі « всесвітньої історії ») / А. І. Федорова // Ст. 76–80.

7. Коваленко Л. Т. Короткий тестологічний словник-довідник / Л. Т. Коваленко. — Київ : Грамота, 2008. — 160 ст.
8. Бабовал Н. Р. (Тернопільський національний економічний університет) Освітні вимірювання в контексті підвищення якості освіти в україні / Н. Р. (Тернопільський національний економічний університет) Бабовал // ІННОВАЦІЙНА ЕКОНОМІКА. — 2016. — Т. 7–8, No. 64. — Ст. 58–63.
9. Артюшина М. В. Психологія діяльності та навчальний менеджмент / М. В. Артюшина, Л. М. Журавська, Л. А. Колесніченко[та ін.]. — Київ : ДВНЗ «Київ. нац. екон. ун-т ім. В.Гетьмана», 2008. — 329 ст.
10. Кобальчинська Є. А. Сучасні комп'ютерні технології тестування знань в навчальному процесі / Є. А. Кобальчинська // ШЛЯХИ РЕАЛІЗАЦІЇ КРЕДИТНО-МОДУЛЬНОЇ СИСТЕМИ ОРГАНІЗАЦІЇ НАВЧАЛЬНОГО ПРОЦЕСУ І ТЕСТОВИХ ФОРМ КОНТРОЛЮ ЗНАНЬ СТУДЕНТІВ. — 2010. — Т. 4. — Ст. 25–29.
11. Цикин И. А. Подготовка и проведение учебных курсов в заочно-дистанционной форме обучения. методические рекомендации преподавателям. / И. А. Цикин. — Санкт-Петербург : СПбГТУ, 2000.
12. Романов А. В. Методика подготовки и проведения тестового контроля в учебном процессе / А. В. Романов. — Чебоксары : «Клио, 1998. — 14-15 ст.
13. Федорук П. І. Адаптивна система дистанційного навчання та контролю знань на базі інтелектуальних інтернет-технологій / П. І. Федорук. — Івано-Франківськ : Видавничо-дизайнерський відділ ЦІТ Прикарпатського національного університету ім. Василя Стефаника, 2008.
14. Цензура М. О., РОЗРОБКА ТЕСТОВИХ ОБОЛОНОК У СЕРЕДОВИЩІ MACROMEDIA AUTHORWARE – ВТЕІ КНТЕУ – // Застосування системи

автоматизованого опитування студентів ВНЗ. – матеріали міжвузівського вебінару – 15.12.2015р.

15. Яремко С.А., Бондар М.В., ВПРОВАДЖЕННЯ ЕЛЕМЕНТІВ СИСТЕМИ УПРАВЛІННЯ НАВЧАННЯМ MOODLE ДЛЯ ПРОВЕДЕННЯ ЕЛЕКТРОННОГО ОПИТУВАННЯ – ВТЕІ КНТЕУ – // Застосування системи автоматизованого опитування студентів ВНЗ. – матеріали міжвузівського вебінару – 15.12.2015р.
16. Система електронного навчання ВНЗ на базі MOODLE : методичний посібник / Ю. В. Триус, І. В. Герасименко, В. М. Франчук ; За ред. Ю. В. Триуса. – Черкаси : ЧДТУ, 2012. – 220 с
17. Штифорук Д.С., АВТОМАТИЗАЦІЯ ПРОВЕДЕННЯ ТЕСТУВАННЯ ЗА ДОПОМОГОЮ ІНТЕРНЕТ РЕСУРСІВ – ВДПУ КНТЕУ – // Застосування системи автоматизованого опитування студентів ВНЗ. – матеріали міжвузівського вебінару – 15.12.2015р.
18. SurveyMonkey. [Електронний ресурс]. URL: https://surveymonkey.com/pricing/individual/?ut_source=pricing-teams-summary
19. Hellip. SurveyMonkey. [Електронний ресурс]. URL: https://hellip.com/en/product/surveymonkey.html
20. roi4cio. SurveyMonkey. [Електронний ресурс]. URL: https://roi4cio.com/catalog/product/surveymonkey
21. Дзюба Т.А., МОЖЛИВОСТІ СИСТЕМИ АВТОМАТИЗОВАНОГО ОПИТУВАННЯ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ ВНЗ – ВТЕІ КНТЕУ – // Застосування системи автоматизованого опитування студентів ВНЗ. – матеріали міжвузівського вебінару – 15.12.2015р.
22. Коваленко О.О., Паламарчук Є.А., ДОСВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ МОДУЛЯ АВТОМАТИЗОВАНОЇ ОЦІНКИ ЗНАНЬ

СТУДЕНТІВ – ВНТУ – // Застосування системи автоматизованого опитування студентів ВНЗ. – матеріали міжвузівського вебінару – 15.12.2015р.

23. Технічна документація Майкрософт. Загальні відомості про адаптивні картки. [Електронний ресурс]. URL: https://docs.microsoft.com/ru-ru/adaptive-cards/
24. Microsoft technical documentation. Understand Actionable Messages in Outlook fundamentals. [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/learn/modules/understand-actionable-messages/
25. Email on Acid. Email Developmen. How to Use Actionable Messages for Outlook 365. [Електронний ресурс]. URL: https://www.emailonacid.com/blog/article/emaildevelopment/how-to-use-actionable-messages-outlook/
26. Навчальні матеріали Microsoft. Tour of Azure services [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/training/modules/intro-to-azure-fundamentals/tour-of-azure-services
27. Microsoft Documentation. Огляд сервісу додатків Azure. [Електронний ресурс]. URL: https://learn.microsoft.com/ru-ru/azure/app-service/overview
28. Microsoft Documentation. Azure Functions documentation. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/azure/azure-functions/
29. Microsoft Documentation. Introduction to Azure Functions. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview
30. Stackshare. Azure App Service vs Azure Functions. [Електронний ресурс]. URL: https://stackshare.io/stackups/azure-app-service-vs-azure-functions

31. Немчинський С. 9 причин вивчити мову C#. //foxminded – 18.07.2022 – [Електронний ресурс]. URL: https://foxminded.ua/9-prichin-vivchiti-movu-c/
32. metanit.com. Введення в C#: Мова C# та платформа .NET. [Електронний ресурс]. URL: https://metanit.com/sharp/tutorial/1.1.php
33. ITVDN forum. Урок 1. Переваги мови C#. [Електронний ресурс]. URL: https://forum.itvdn.com/t/urok-1-preimushhestva-yazyka-c/2941
34. Вікіпедія. .NET. [Електронний ресурс]. URL: https://ru.wikipedia.org/wiki/.NET
35. Microsoft Documentation. What's new in .NET 6. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-6
36. Zmerzlyi I., Мікросервісна архітектура. Medium. [Електронний ресурс]. URL: https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d
37. Kravchuk A. Мікросервісна архітектура: плюси та мінуси. ITEDU//BLOG. [Електронний ресурс]. URL: https://blog.iteducenter.ua/articles/microservices-architecture-advantages-and-disadventages/
38. Internetdevels. Building microservices. [Електронний ресурс]. URL: https://internetdevels.ua/blog/building-microservices
39. Хабр. Налаштовуємо свою кімнатну Service Bus for Windows Server. – 26.03.2015 – [Електронний ресурс]. URL: https://habr.com/ru/post/254059/
40. Microsoft Documentation. What is Azure Service Bus? [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-messaging-overview
41. Microsoft Documentation. Message sequencing and timestamps. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/azure/service-bus-messaging/message-sequencing#scheduled-messages

42. pmichaels.net. Azure Service Bus - Scheduled Message Delivery. – 01.01.2021 – [Електронний ресурс]. URL: https://pmichaels.net/2021/01/01/azure-service-bus-scheduled-message-delivery/
43. Serverless notes. Schedule messages on your queue. – 09.09.2022 – [Електронний ресурс]. URL: https://www.serverlessnotes.com/docs/schedule-messages-on-your-queue
44. METANIT.COM. Сайт про програмування. Введення в Blazor. [Електронний ресурс]. URL: https://metanit.com/sharp/blazor/1.1.php
45. Mobilesop. СИСТЕМА АНАЛІТИКИ ДАНИХ. [Електронний ресурс]. URL: https://ua.mobilesop.com/resheniya/sistema-analitiki-dannyih/
46. Favour D., Ankit A., Kibana vs Grafana - Which tool to choose?. SigNoz. – 22.06.2022 – [Електронний ресурс]. URL: https://signoz.io/blog/kibana-vs-grafana/
47. Microsoft Documentation. Application Insights overview. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview?tabs=net
48. APIXDrive. Integration SendGrid. Електронний ресурс]. URL: https://apix-drive.com/en/sendgrid
49. Startpack. SendGrid. Електронний ресурс]. URL: https://startpack.ru/application/sendgrid#features
50. Горбачова Е.О., Ковалюк О.О., Розробка чат-боту для автоматизації навчального процесу – ВНТУ – //бакалаврська кваліфікаційна робота – 2021р.
51. Git [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Git
52. Azure DevOps Server [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Azure_DevOps_Server

53. What is PowerShell? [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.1
54. Microsoft Documentation. What is Bicep?. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/training/modules/introduction-to-infrastructure-as-code-using-bicep/4-what-bicep
55. EPAM blog. Мова Bicep в Azure Cloud та інтеграція Amazon API Gateway. – 07.11.2022. – [Електронний ресурс]. URL: https://careers.epam.ua/blog/bicep-language-in-azure-cloud-and-amazon-api-gateway-integration
56. Microsoft Documentation. How Bicep works. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/training/modules/introduction-to-infrastructure-as-code-using-bicep/5-how-bicep-works
57. Microsoft Documentation. Refresh an actionable message when the user opens it. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/outlook/actionable-messages/auto-invoke
58. Wikipedia. Inversion of control. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/outlook/actionable-messages/auto-invoke

ДОДАТКИ

Додаток А
(Обов'язковий)
ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри КСУ ВНТУ,
д.т.н., доцент
_____В'ячеслав КОВТУН
« 14 » 12 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Розробка інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу
08-33.МКР.001.00.000 ТЗ

Студент групи 2АКІТ-21м
_____ Евеліна ГОРБАЧОВА
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник к.т.н., доцент
_____ Олег КОВАЛЮК
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2022

1. Назва та галузь застосування

1.1. Назва – Розробка інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу.

1.2. Галузь застосування – освіта.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від "14" вересня 2022 року №203

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи ефективності оцінювання якості знань студентів, використовуючи автоматизованої системи інтерактивних поштових електронних повідомлень.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Горбачова Е.О, Ковалюк О.О., Розробка інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу// ВНТУ. – 2022. – [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2022/paper/view/14206
2. ГОРБАТЮК Р. Тестування як метод педагогічного контролю якості знань майбутніх фахівців із вищою освітою / Р. ГОРБАТЮК, Т. СІТКАР // Нова педагогічна думка. — 2014. — Т. 3, No. 3. — Ст. 179–183.
3. Microsoft. Actionable Message Designer. [Електронний ресурс]. URL: https://amdesigner.azurewebsites.net/

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- Можливість створювати і змінювати форми опитування.
- Можливість встановлювати час відправки.
- Функція відправки електроних повідомлень.
- Відображення статистики реазультатів.
- Автоматичне розгортування системи.
- Система авторизації на базі Microsoft Identity.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10;
- Доступ до інтернету;
- Наявна електронна пошта;
- Аккаунт в Azure Devops;

5.2.2. Умови експлуатації системи:

- робота на стандартних ПЕОМ та мобільних телефонах в приміщеннях зі стандартними умовами;
- можливість цілодобового функціонування системи;
- текст програмного забезпечення системи є цілком закритим.
- дані, що зберігаються є захищеними.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів, принципів, підходів і засобів реалізації задачі автоматизації процесами в об’єкті управління відповідно до теми кваліфікаційної роботи. Постановка задач дослідження «03»_09__ 2022 р.
2. Багатоваріантний аналіз сучасних технологій для побудови опитувань та можливостей обробки результатів «_19_» __09__ 2021 р.
3. Обґрунтування переваг використання Actionable Messages «_25_»_09_ 2022 р.
4. Побудова архітектури додатку «_3_» _10____ 2022 р.
5. Розробка архітектури розгортання додатку «_5_» _10___ 2022 р.
6. Розробка головного додатку «_17_» __10___ 2022 р.
7. Розробка Devops частини «_23_» __11___ 2022 р.

6.2 Графічні матеріали:

1. Розробка UML-діаграм системи «_5_» _12___ 2022 р.
2. Тестування програмного забезпечення «_5_» __12__ 2022 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «28»__11_ 2022 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «16»__12_ 2022 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «23»__12_ 2022 р.

Додаток Б
(Обов'язковий)

ПРОТОКОЛ

ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка інтерактивних форм в поштових електронних повідомленнях для автоматизації навчального процесу»

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 84,6% Схожість 15,4%

Аналіз звіту подібності (відмітити потрібне)

☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Галущак А.В.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Горбачова Е.О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Ковалюк О.О.
(підпис) (прізвище, ініціали)

Додаток В
(Довідниковий)

Лістинг програми

Клас AdaptiveCardTemplateBuilder:

```
using System.Collections.Generic;
using System.Linq;
using Newtonsoft.Json;
using vntu.study.notification.Components.DataTransferObjects.Incomming;
using Vntu.Study.Notification.Components.DataTransferObjects.Internal;

namespace vntu.study.notification.AdaptiveCards
{
    public interface IAdaptiveCardTemplateBuilder
    {
        string BuildQuestion(int questionNumber, SendEmailMessageQuestions ques
tion);
        string BuildSubmit(IEnumerable<SendEmailMessageQuestions> questions);
        string BuildAdaptiveCard(string questionSection, string submitBody, str
ing submitUrl, string refreshUrl);
        string BuildHtmlMessage(string adaptiveMessagePayload);
        string ReplaseSection(string template, string sectionName, string secti
onContent);
        IEnumerable<string> BuildAnswers(IEnumerable<SendEmailMessageAnswers> a
nswers);
    }

    public sealed class AdaptiveCardTemplateBuilder : IAdaptiveCardTemplateBuil
der
    {
        private readonly IAdaptiveCardTemplateProvider templateProvider;

        public const string AnswerTemplateSectionName = "answerSection";
        public const string QuestionTemplateSectionName = "questionsSection";
        public const string ScriptTemplateSectionName = "scriptSection";

        public AdaptiveCardTemplateBuilder(IAdaptiveCardTemplateProvider templa
teProvider)
        {
            this.templateProvider = templateProvider;
        }

        public string BuildQuestion(int questionNumber, SendEmailMessageQuestio
ns question) =>
            this.templateProvider.QuestionMessageCard
                .Replace("{{questionNumber}}", questionNumber.ToString())
                .Replace("{{question}}", question.Question)
```

```
                .Replace($@"{{{{""questionType""}}}}", question.Type == Quesnti
onType.Single ? "false" : "true");

        public IEnumerable<string> BuildAnswers(IEnumerable<SendEmailMessageAns
wers> answers)
        {
            string template = this.templateProvider.AnswerMessageCard;

            return answers.Select((e, i) =>
            {
                string answerContent = template
                                       .Replace("{{title}}", e.Answer)
                                       .Replace("{{number}}", i.ToString());

                return answerContent;
            });
        }

        public string BuildSubmit(IEnumerable<SendEmailMessageQuestions> questi
ons)
        {
            var submitTypeQuestion = questions.Select((e, i) => new SubmitCardP
ayload
            {
                Question = e.Question,
                Answer = $"{{{{question{i}.value}}}}"
            });

            return JsonConvert.SerializeObject(submitTypeQuestion);
        }

        public string BuildAdaptiveCard(string questionSection, string submitBo
dy, string submitUrl, string refreshUrl)
        {
            string template = this.templateProvider.AdaptiveCardTemplate;

            template = this.ReplaseSection(template, questionSection, QuestionT
emplateSectionName);
            template.Replace("{{submitBody}}", submitBody);
            template.Replace("{{surveySubmitUrl}}", submitUrl);
            template.Replace("{{refreshUrl}}", refreshUrl);

            return template;
        }

        public string BuildHtmlMessage(string adaptiveMessagePayload)
        {
            var template = this.templateProvider.AdaptiveHtmlTemplate;

            template = this.ReplaseSection(template, adaptiveMessagePayload, Sc
riptTemplateSectionName);
```

```
            return template;
        }

        public string ReplaseSection(string template, string sectionContent, st
ring sectionName) => template.Replace($@"[[ ""{sectionName}"" ]]", sectionConte
nt);
    }
}
```

Клас AdaptiveCardTemplateProvider:

```
using System;
using System.IO;
using System.Reflection;

namespace vntu.study.notification.AdaptiveCards
{
    public interface IAdaptiveCardTemplateProvider
    {
        string AdaptiveHtmlTemplate { get; }
        string AdaptiveCardTemplate { get; }
        string AnswerMessageCard { get; }
        string ErrorMessageCard { get; }
        string QuestionMessageCard { get; }
    }

    public sealed class AdaptiveCardTemplateProvider : IAdaptiveCardTemplatePro
vider
    {
        private string emailLayout { get; set; }
        private string cardLayout { get; set; }
        private string questionMessageCard { get; set; }
        private string answerMessageCard { get; set; }
        private string errorTemplate { get; set; }

        public string AdaptiveHtmlTemplate
        {
            get
            {
                if (emailLayout == null)
                {
                    emailLayout = ReadFromFile("emailtemplate.html");
                }
                return new string(emailLayout);
            }
        }
        public string AdaptiveCardTemplate
        {
            get
            {
```

```
                if (cardLayout == null)
                {
                    cardLayout = ReadFromFile("cardTemplateLayout.json");
                }
                return new string(cardLayout);
            }
        }

        public string AnswerMessageCard
        {
            get
            {
                if (answerMessageCard == null)
                {
                    answerMessageCard = ReadFromFile(Path.Combine("Questions",
"answerTemplate.json"));
                }
                return new string(answerMessageCard);
            }
        }

        public string QuestionMessageCard
        {
            get
            {
                if (questionMessageCard == null)
                {
                    questionMessageCard = ReadFromFile(Path.Combine("Questions"
, "questionTemplate.json"));
                }
                return new string(questionMessageCard);
            }
        }

        public string ErrorMessageCard
        {
            get
            {
                if (errorTemplate == null)
                {
                    errorTemplate = ReadFromFile("errorMessageCard.json");
                }
                return new string(errorTemplate);
            }
        }

        private static string ReadFromFile(string fileTemplateName)
        {
            string homeAppAPath = Environment.GetEnvironmentVariable("HOME");
            string functionRootPath = homeAppAPath == null ? Environment.Curren
tDirectory : Path.Combine(homeAppAPath, "site", "wwwroot");
```

```
            string[] templateFolderPath = { functionRootPath, "AdaptiveCards",
"Templates", fileTemplateName };

            // Files that are located in the functions solution
            if (File.Exists(Path.Combine(templateFolderPath)))
            {
                return File.ReadAllText(Path.Combine(templateFolderPath));
            }
            else
            {
                // Files that are located in the web app solutions
                string assemblyFolder = Path.GetDirectoryName(Assembly.GetExecu
tingAssembly().Location);
                string jsonPath = Path.Combine(assemblyFolder, "AdaptiveCards",
 fileTemplateName);

                return File.ReadAllText(Path.Combine(jsonPath));
               }
          }
     }
}
```

Код інтеграції Send Grid з системою. SendGridApiService:

```
using System.Threading.Tasks;
using SendGrid;
using SendGrid.Helpers.Mail;

namespace vntu.study.notification.Infrastructure
{
    public interface ISendGridApiService
    {
        Task SendAsync(string messageBody, string recieverEmail, string subject
Name);
    }

    public sealed class SendGridApiService : ISendGridApiService
    {
        private readonly string SendGridApiKey = "SG.RZiwRA_hTt2wMxkgGqqLgg.j-
Cx-1ynPmLPyGiwfGpCzs4S7Hu5oAioCweSmMmlEuA";
        private readonly string FromEmail = "notification@evelinamsazoutlook.on
microsoft.com";

        public async Task SendAsync(string messageBody, string recieverEmail, s
tring subjectName)
        {
            var client = new SendGridClient(SendGridApiKey);

            var fromEmail = new EmailAddress(FromEmail);
            var toEmail = new EmailAddress(recieverEmail);
```

```
            var subject = "Vntu Test";
            var htmlContent = messageBody;

            var msg = MailHelper.CreateSingleEmail(fromEmail, toEmail, subject,
 string.Empty, htmlContent);
            var _ = await client.SendEmailAsync(msg).ConfigureAwait(false);
        }
    }
}
```

Storage Account Client:

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Azure;
using Azure.Data.Tables;

namespace Vntu.Study.Notification.Infrastructure.StorageAccount
{
    public interface IStorageAccountClient
    {
        void Add<T>(T entity) where T : ITableEntity;
        Task<T> Get<T>(string rowKey, string partitionKey) where T : class, ITa
bleEntity, new();
        Task<IEnumerable<T>> GetList<T>(string partitionKay) where T : class, I
TableEntity, new();
    }

    public sealed class StorageAccountClient : IStorageAccountClient
    {
        private readonly TableClient tableClient;

        public StorageAccountClient(TableClient tableClient)
        {
            this.tableClient = tableClient;
        }

        public void Add<T>(T entity) where T : ITableEntity
        {
            this.tableClient.CreateIfNotExists();
            this.tableClient.AddEntity(entity);
        }

        public async Task<T> Get<T>(string rowKey, string partitionKey) where T
 : class, ITableEntity, new()
        {
            var tableResponse = await this.tableClient.GetEntityAsync<T>(partit
ionKey, rowKey);
            return tableResponse.Value;
        }
```

```
        public async Task<IEnumerable<T>> GetList<T>(string partitionKay) where
 T : class, ITableEntity, new()
        {
            var fullList = new List<T>();

            AsyncPageable<T> queryResult = this.tableClient.QueryAsync<T>(e =>
e.PartitionKey.Equals(partitionKay, System.StringComparison.OrdinalIgnoreCase))
;
            await foreach (Page<T> page in queryResult.AsPages())
            {
                foreach (T qEntity in page.Values)
                {
                    fullList.Add(qEntity);
                }
            }

            return fullList;
        }
    }
}
```

Приклад сторінок написаних на Blazor:

```
@page "/test"
@attribute [Authorize]


@using Microsoft.AspNetCore.Authorization;
@using vntu.study.webapp.Models


<h3>Todo</h3>


<div class="input-group mb-3">
   <div class="input-group-prepend">
      <span class="input-group-text" id="inputGroup-sizing-default">Назва
тесту</span>
   </div>
   <input @bind="Subject" class="form-control" aria-label="Default" aria-
describedby="inputGroup-sizing-default">
```

```
</div>

<div class="input-group mb-3">
   <div class="input-group-prepend">
      <span class="input-group-text" id="inputGroup-sizing-
default">Питання</span>
   </div>
   <input @bind="Question" type="text" class="form-control" aria-
label="Default" aria-describedby="inputGroup-sizing-default">
</div>

@foreach (var answer in answers)
{
   <div class="row">
      <div class="input-group mb-3">
         <div class="input-group-text">
            <input @bind="answer.IsCorrect" name="contact" class="form-
check-input mt-0" type="radio" aria-label="Radio button for following text
input">
         </div>
         <input @bind="answer.Answer" type="text" class="form-control" aria-
label="Text input with radio button" />
      </div>
   </div>
}

<div class="row row-cols-auto">
   <div class="col">
      <button type="button mb-3" class="btn btn-primary"
@onclick="AddAnswer">Додати відповідь</button>
```

```
    </div>
    <div class="col">
        <button type="button mb-3" class="btn btn-secondary"
@onclick="AddQuestion">Додати питання</button>
    </div>
    <div class="col">
        <button type="button mb-3" class="btn btn-success"
@onclick="AddAnswer">Зберегти питання</button>
    </div>
</div>

<div class="row">
    @foreach (var question in questions)
    {
        <div>@question.Question</div>
        foreach (var answer in question.Answers)
        {
            <div>@answer.Answer @answer.IsCorrect</div>
        }
    }
</div>


@code {
    private string Subject;
    private string Question;

    private List<TestQuestionAnswer> answers = new();
    private List<TestQuestion> questions = new();
```

```
        private void AddAnswer() => answers.Add(new TestQuestionAnswer());
        private void AddQuestion()
        {
            questions.Add(new TestQuestion
                {
                    Question = new string(Question),
                    Answers = new List<TestQuestionAnswer>(answers)
                });

            answers.Clear();
            Question = string.Empty;
        }
}
```

Сервіс відправки повідомлень. SendCardService :

```
using System;
using System.Linq;
using System.Threading.Tasks;
using vntu.study.notification.AdaptiveCards;
using vntu.study.notification.Components.DataTransferObjects.Incomming;
using vntu.study.notification.Infrastructure;
using Vntu.Study.Notification.Components.DataTransferObjects.Internal;
using Vntu.Study.Notification.Infrastructure.Exntensions;
using Vntu.Study.Notification.Infrastructure.StorageAccount;

namespace vntu.study.notification.Services.SendCard
{
    public interface ISendCardService
    {
        Task SendAsync(SendEmailMessage message);
    }

    public sealed class SendCardService : ISendCardService
    {
        private readonly IAdaptiveCardTemplateBuilder templateBuilder;
        private readonly IStorageAccountClient storageAccountClient;
        private readonly ISendGridApiService sendGridApiService;

        public SendCardService(
            IAdaptiveCardTemplateBuilder templateBuilder,
            IStorageAccountClient storageAccountClient,
```

```
            ISendGridApiService sendGridApiService)
        {
            this.templateBuilder = templateBuilder;
            this.storageAccountClient = storageAccountClient;
            this.sendGridApiService = sendGridApiService;
        }

        public async Task SendAsync(SendEmailMessage message)
        {
            int questionPossition = 0;

            var questionsSections = message.Questions.Select(question =>
            {
                string questionString = this.templateBuilder.BuildQuestion(ques
tionPossition++, question).EscapeRootArray();
                var answersStrings = this.templateBuilder.BuildAnswers(question
.Answers).Select(x => x.EscapeRootArray());

                string questionTemplate = this.templateBuilder.ReplaseSection(
                    questionString, string.Join(",", answersStrings),
                    AdaptiveCardTemplateBuilder.AnswerTemplateSectionName);

                return questionTemplate.EscapeRootArray();
            });

            string adaptiveCard = this.templateBuilder.BuildAdaptiveCard(string
.Join(",", questionsSections), "", "", "");
            string htmlTemplate = this.templateBuilder.BuildHtmlMessage(adaptiv
eCard);

            var adaptiveCardState = new AdaptiveCardState
            {
                Template = adaptiveCard,
                CardId = Guid.NewGuid().ToString(),
                TeachedId = message.TeacherId,
                StudentId = message.SendToEmail
            };

            this.storageAccountClient.Add(adaptiveCardState);
            await this.sendGridApiService.SendAsync(htmlTemplate, message.SendT
oEmail, message.SubjectName);
        }
    }
}
```

Додаток Г
(Обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ІНТЕРАКТИВНИХ ФОРМ В ПОШТОВИХ ЕЛЕКТРОННИХ ПОВІДОМЛЕННЯХ ДЛЯ АВТОМАТИЗАЦІЇ НАВЧАЛЬНОГО ПРОЦЕСУ

1. Схема архітектури розробки
3. Схема розгортання ресурсів та доставка коду
4. Мапа використовуваних ресурсів Azure cloud
5. Структура класів та зв'язків у сервісах, що відповідають за створення та управління адаптивними картками
6. Структура класів та зв'язків у сервісах, що відповідають за обробку та управління адаптивними картками для веб сервісу
7. UML діаграма послідовності роботи системи
8. Результати тестування. Зображення інтерфейсу порталу викладача: сторінка створення тесту
9. Результати тестування. Зображення прикладу створення тесту
10. Результати тестування. Зображення прикладу роботи сайту при створенні тесту
11. Результати тестування. Зображення відправленого тесту та строки його стану.
12. Результати тестування. Скрін інтерактивної форми із комп'ютера
13. Результати тестування. Скрін інтерактивної форми із телефону
14. Результати тестування. Зображення форми із дизайнера Actionable Message Designer

Студент групи 2АКІТ-21м _____ Евеліна ГОРБАЧОВА
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник к.т.н, доцент _____ Олег КОВАЛЮК
Підпис *Ім'я ПРІЗВИЩЕ*

СХЕМА АРХІТЕКТУРИ РОЗРОБКИ

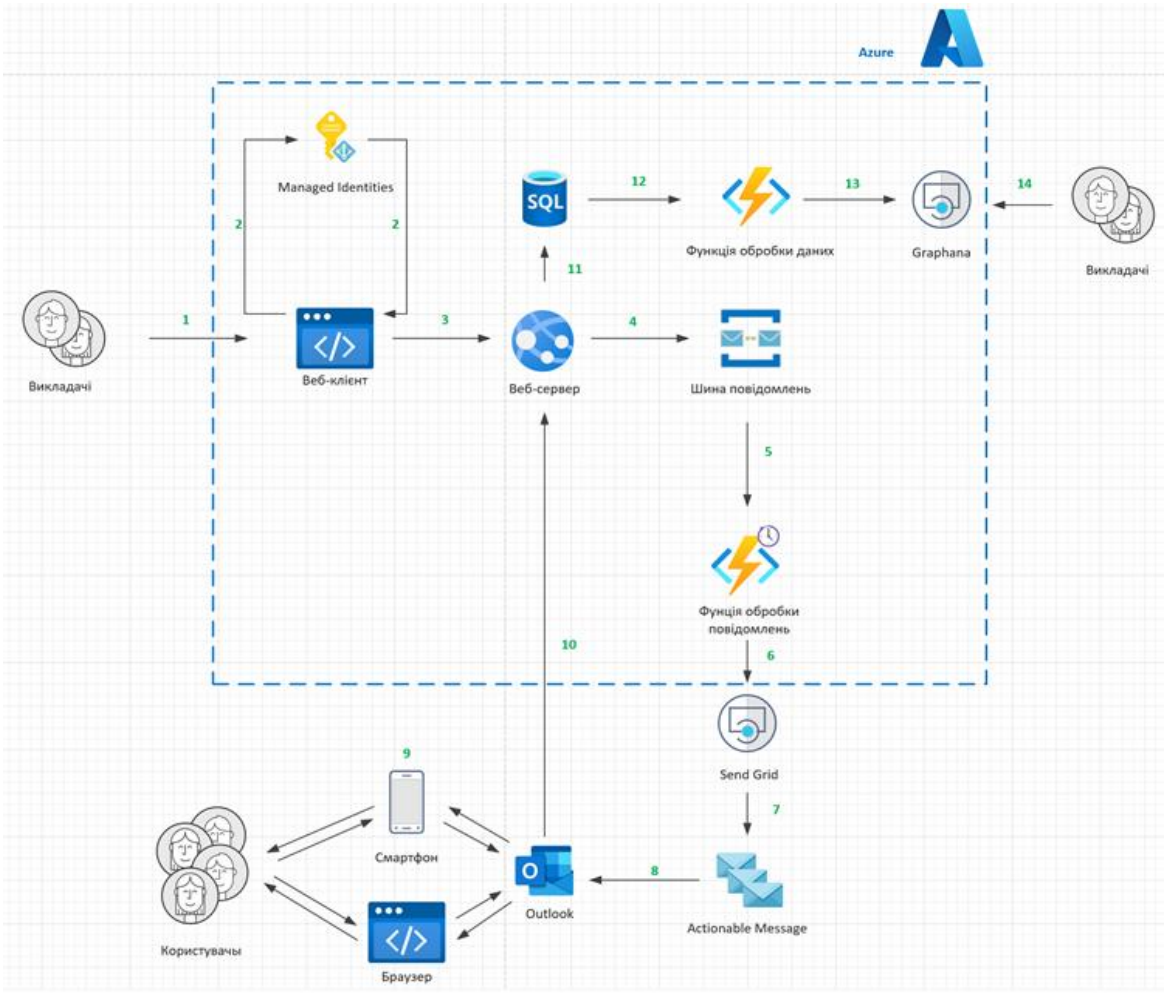
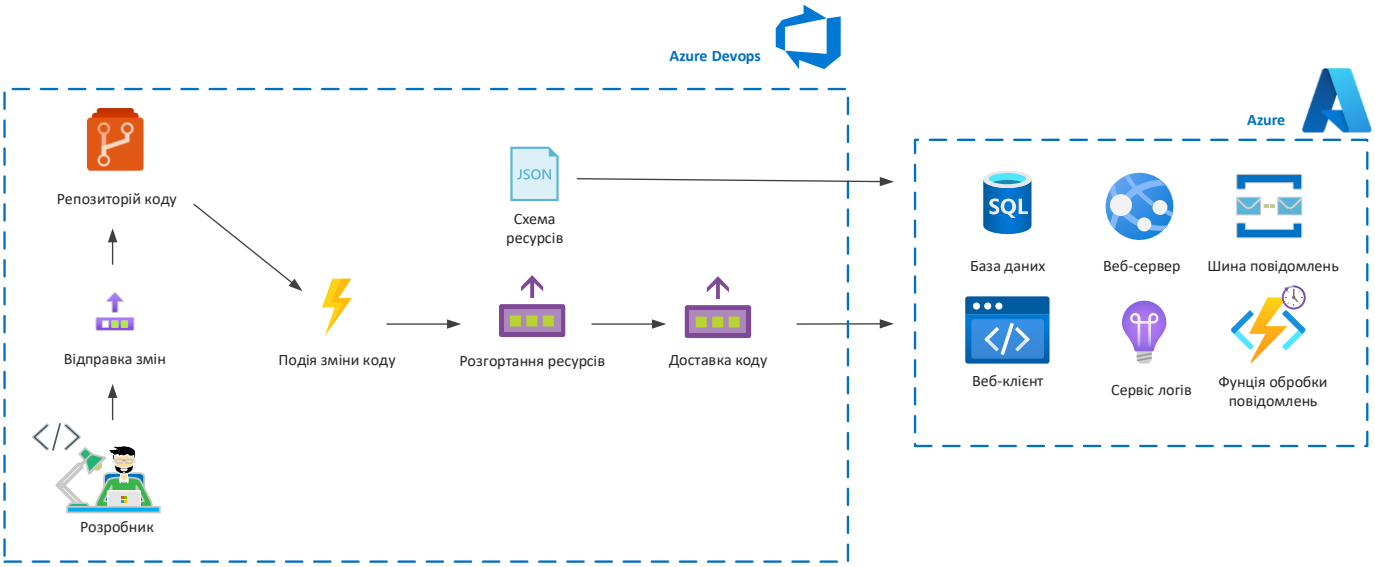
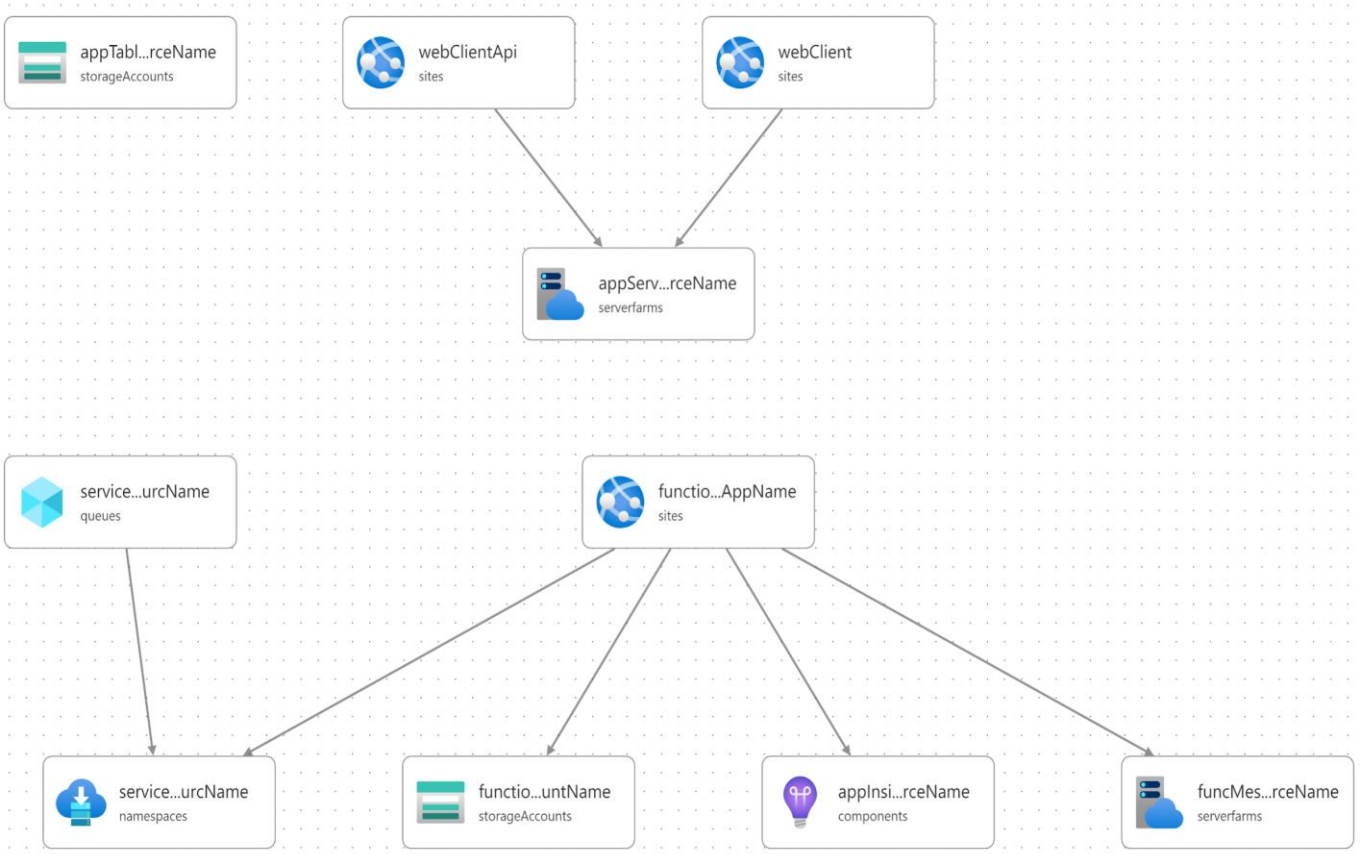


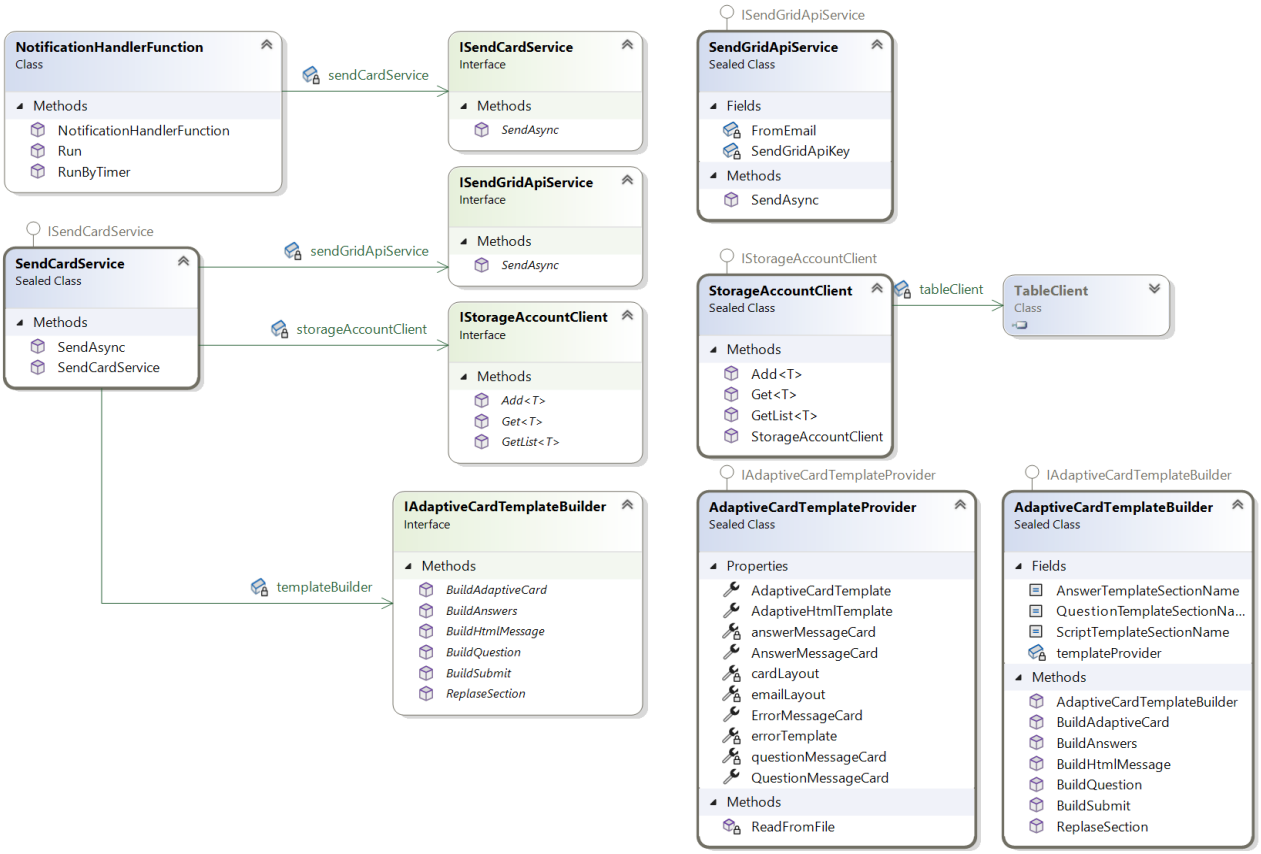
СХЕМА РОЗГОРТАННЯ РЕСУРСІВ ТА ДОСТАВКА КОДУ



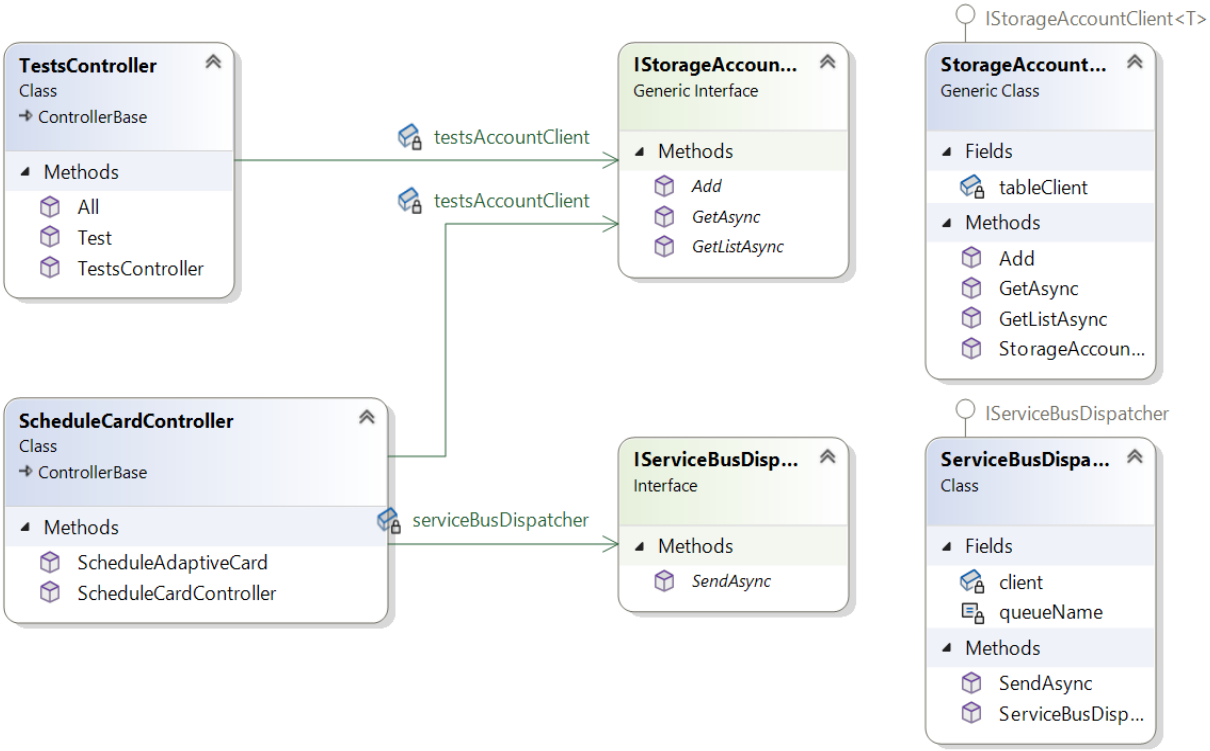
МАПА ВИКОРИСТОВУВАНИХ РЕСУРСІВ AZURE CLOUD



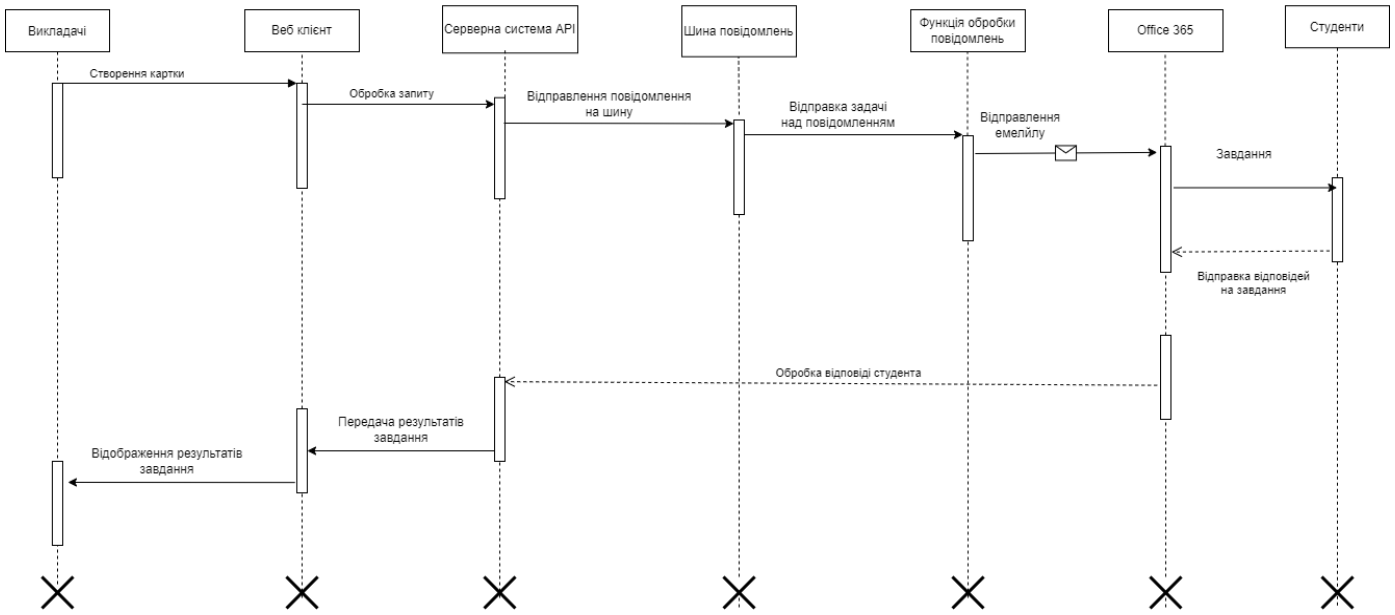
СТРУКТУРА КЛАСІВ ТА ЗВ'ЯЗКІВ У СЕРВІСАХ, ЩО ВІДПОВІДАЮТЬ ЗА СТВОРЕННЯ ТА УПРАВЛІННЯ АДАПТИВНИМИ КАРТКАМИ



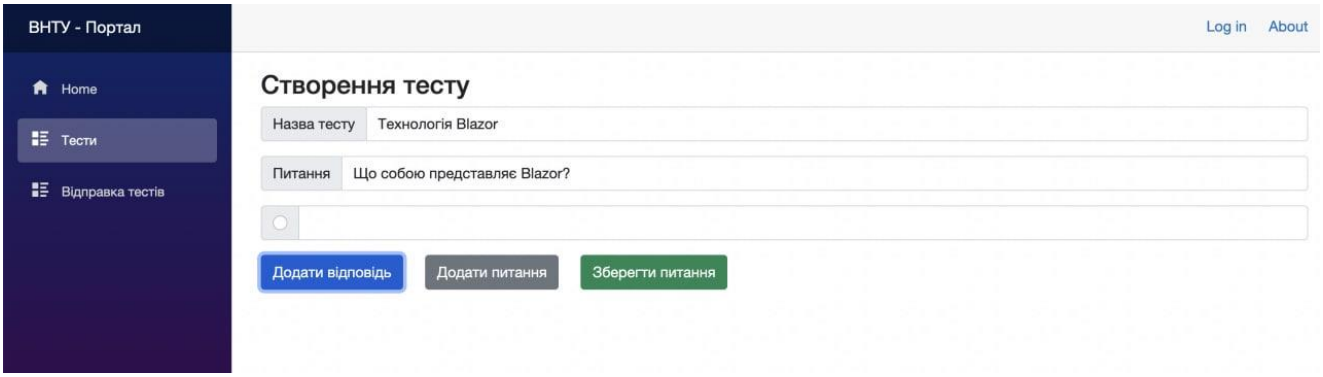
СТРУКТУРА КЛАСІВ ТА ЗВ'ЯЗКІВ У СЕРВІСАХ, ЩО ВІДПОВІДАЮТЬ ЗА ОБРОБКУ ТА УПРАВЛІННЯ АДАПТИВНИМИ КАРТКАМИ ДЛЯ ВЕБ СЕРВІСУ



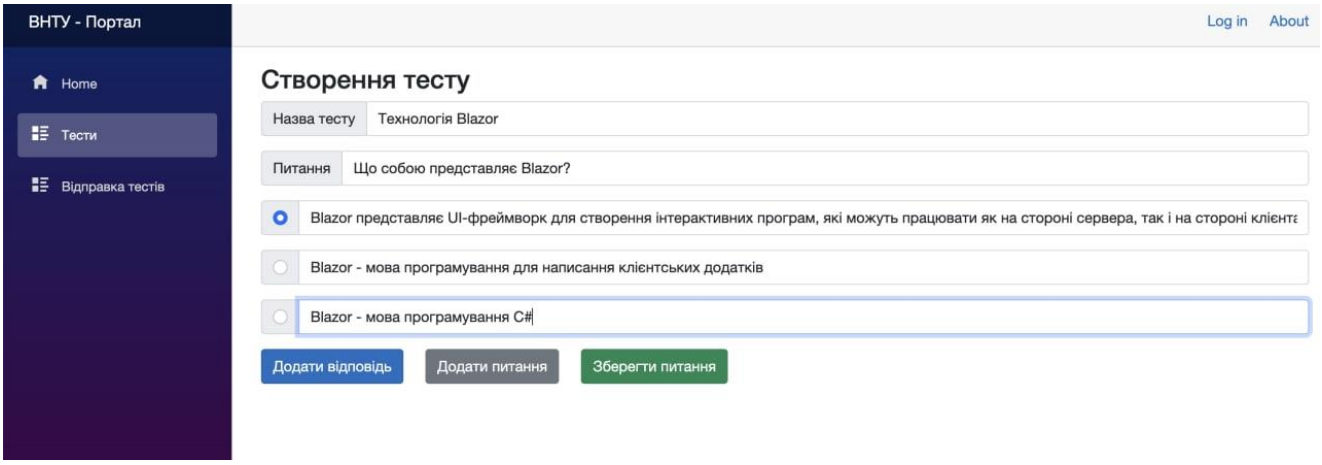
UML ДІАГРАМА ПОСЛІДОВНОСТІ РОБОТИ СИСТЕМИ



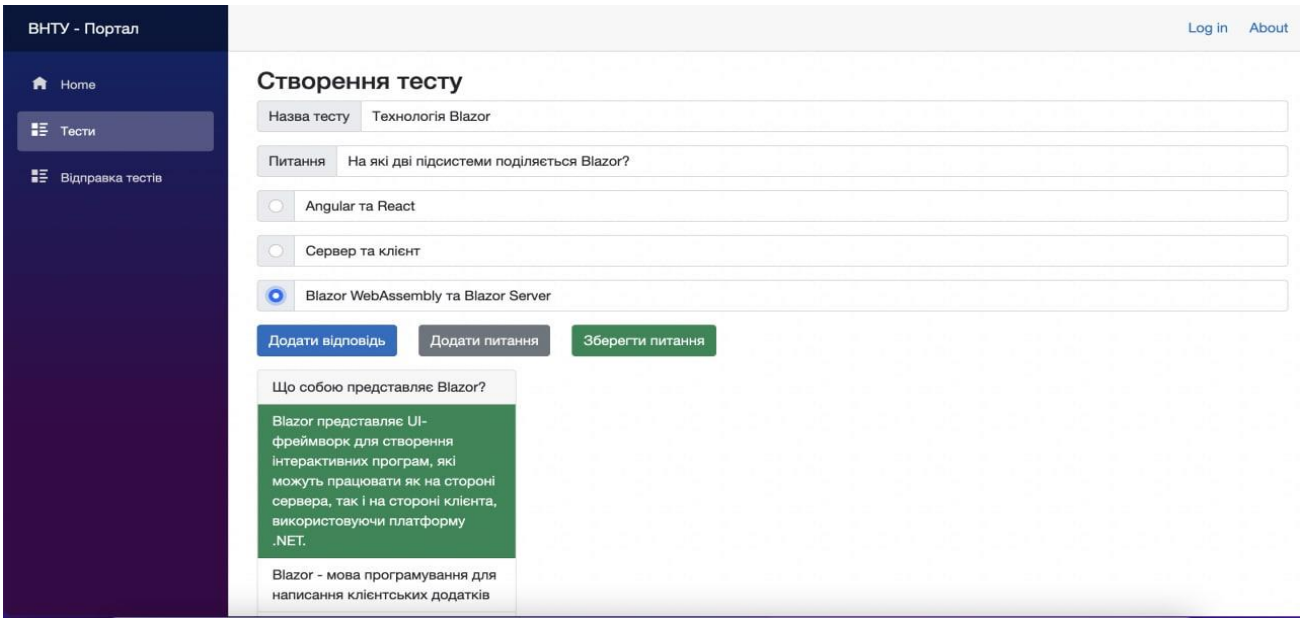
РЕЗУЛЬТАТИ ТЕСТУВАННЯ



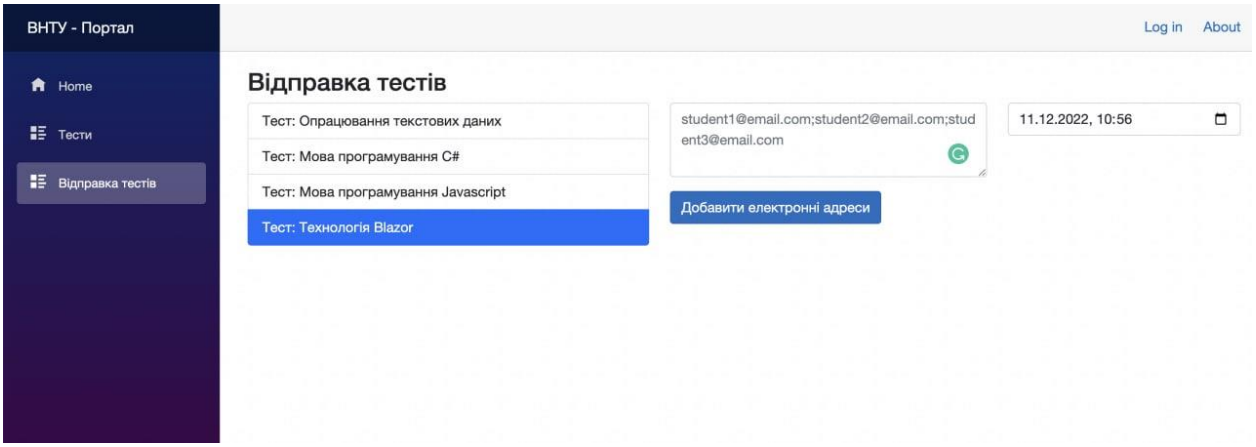
Зображення інтерфейсу порталу викладача: сторінка створення тесту



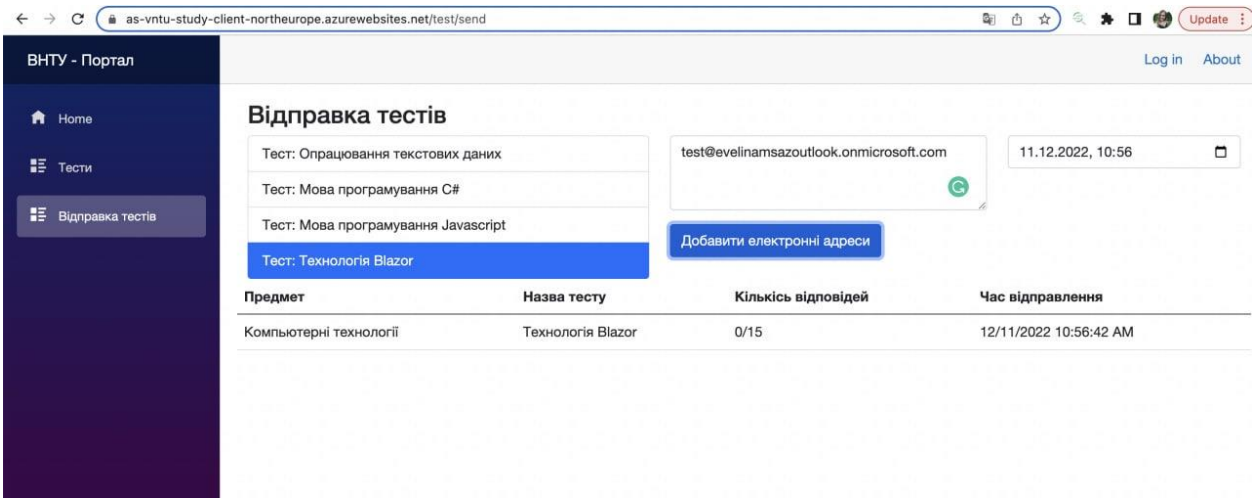
Зображення прикладу створення тесту



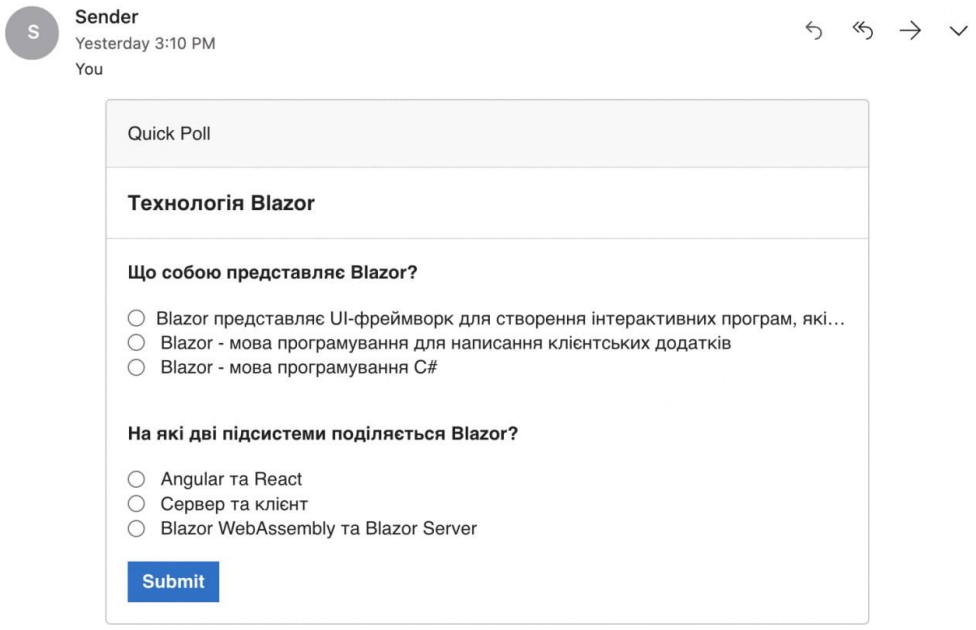
Зображення прикладу роботи сайту при створенні тесту



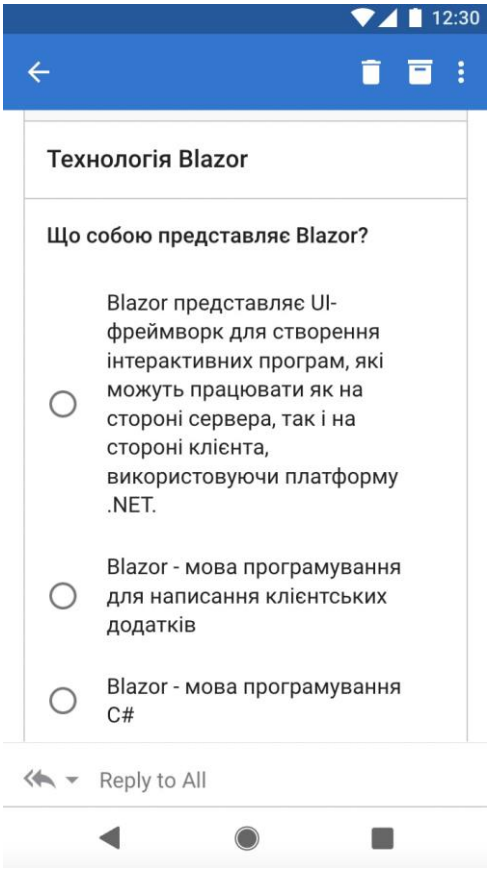
Зображення сторінки «відправки тестів»



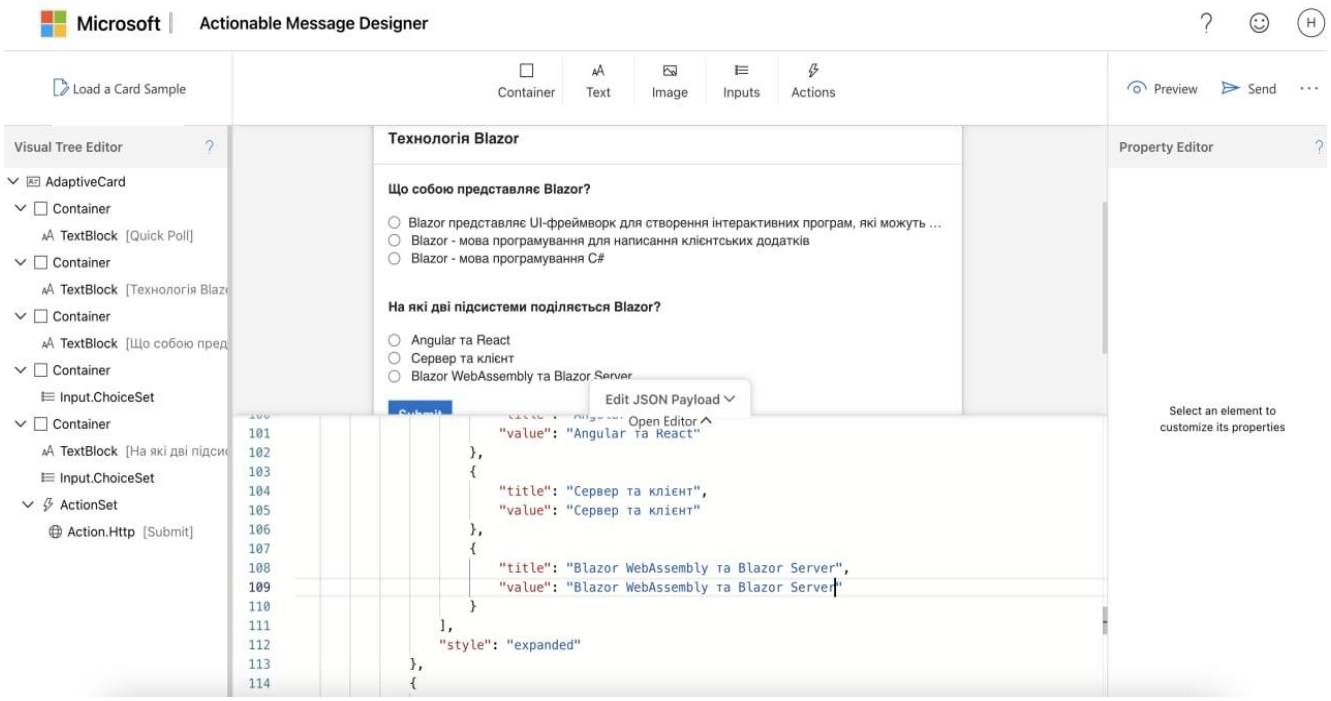
Зображення відправленого тесту та строки його стану.



Скрін інтерактивної форми із комп’ютера



Скрін інтерактивної форми із телефону



Зображення форми із дизайнера Actionable Message Designer