Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

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

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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Програмна система оптимізації створення голосового меню IVR»

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

Керівник: к.т.н., доцент кафедри ПЗ
Рейда О.М.
(прізвище та ініціали)
«____»_____2021 р.

Опонент:
д.т.н., проф. Василевський О. М (прізвище та ініціали)
«____»_____2021 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«____»_____2021 р.

Вінниця ВНТУ - 2021 рік

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

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

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

Лічману Едуарду Руслановичу

1. Тема роботи – Програмна система оптимізації створення голосового меню IVR.

Керівник роботи: Рейда О. М., к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 24 » вересня 2021 р. № 277.

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи:

Стандартизоване голосове меню IVR, параметри сценарію IVR, звукові файли із оголошенням пунктів меню IVR.

4. Зміст розрахунково-пояснювальної записки: вступ, аналіз стану питання та постановка задач, розробка структури програмного продукту, розробка програмних засобів, тестування системи, економічна частина.

5. Перелік графічного матеріалу: актуальність обраної теми; основні задачі дослідження; наукова новизна одержаних результатів; порівняльний аналіз аналогів; архітектура системи; лістинг

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Рейда О. М., к.т.н., доцент кафедри ПЗ		
5	Бурєннікова Н. В., д.е.н., проф. кафедри ЕПВМ		

7. Дата видачі завдання 14 вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання та постановка задач дослідження	15.09.2021-26.09.2021	Вик.
2	Розробка структури програмного продукту	27.09.2021-15.10.2021	Вик.
3	Розробка програмних засобів	16.10.2021-7.11.2021	Вик.
4	Тестування системи	8.11.2021-21.11.2021	Вик.
5.	Економічна частина	22.11.2021-30.11.2021	Вик.

Студент _____ **Лічман Е. Р.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Рейда О. М.**
(підпис) (прізвище та ініціали)

Опонент магістерської кваліфікаційної роботи _____ **Василевський О.М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 681.326.3

Лічман Е. Р.: Програмна система оптимізації створення голосового меню IVR. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2021. 115 с.

На укр. мові. Бібліогр.: 30 назв; рис.: 46; табл.: 5.

У магістерській кваліфікаційній роботі проведено детальний аналіз методів і засобів створення голосового меню IVR, обґрунтовано необхідність підвищення їх продуктивності та ефективності.

Розглянуто та проаналізовано існуючі методи та підходи в створенні голосового меню IVR. Запропоновано алгоритм створення голосового меню IVR. На його основі спроектовано та розроблено програмну систему, що дозволяє усунути на етапі створення голосового меню IVR кроки, виконання яких вимагає спеціальних знань від людини, що їх заповнює або є потенційним місцем збою для системи.

Проведено тестування системи на різних вхідних даних та доведено її працездатність та ефективність.

Розроблену систему можна використовувати для зменшення часу створення голосового меню IVR, економії ресурсів, таких як: кошти та час. Також дозволяє збільшити прибуток та лояльність клієнтів.

У процесі досліджень використовувались: методи теорії алгоритмів для розробки алгоритму і програмних модулів, методи контейнеризації, віртуалізації для створення dev-оточення, методи та способи оптимізації даних, методи теорії баз даних для розробки структури бази даних; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Ключові слова: IVR, голосове меню, сценарій.

ABSTRACT

УДК 681.326.3

Lichman ER: Software system for optimizing the creation of IVR voice menu. Master's thesis in specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2021. 115 p.

In Ukrainian language. Bibliographer: 30 titles; fig.: 46; tabl.: 5.

In the master's qualification work the detailed analysis of methods and means of creation of the IVR voice menu is carried out, the necessity of increase of their productivity and efficiency is proved..

The existing methods and approaches in creating an IVR voice menu are considered and analyzed. An algorithm for creating an IVR voice menu is proposed. Based on it, a software system has been designed and developed, which allows eliminating at the stage of creating an IVR voice menu steps that require special knowledge from the person who fills them or is a potential place of failure for the system.

The system was tested on various input data and its efficiency and effectiveness were proved.

The developed system can be used to reduce the time of creating an IVR voice menu, saving resources such as: money and time. It also allows you to increase profits and customer loyalty.

The research used: methods of algorithm theory for algorithm development and software modules, methods of containerization, virtualization to create a dev environment, methods and ways to optimize data, methods of database theory to develop a database structure; computer modeling to analyze and verify the obtained theoretical positions.

Key words: IVR, voice menu, script.

ЗМІСТ

ВСТУП **8**

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ **11**

1.1 Аналіз стану питання 11

1.2 Порівняльний аналіз аналогів 13

1.3 Аналіз алгоритмів створення голосового меню IVR 17

1.4 Постановка задачі розробки 20

1.5 Висновки 20

2 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ПРОДУКТУ 22

2.1 Аналіз методів та засобів розробки веб-додатків 22

2.2 Розробка загальної моделі системи 24

2.3 Розробка діаграми послідовності 33

2.4 Розробка алгоритму створення голосового меню IVR 35

2.5 Висновки 42

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ 43

3.1 Варіантний аналіз і обґрунтування вибору засобів реалізації програмного засобу 43

3.2 Розробка API сервісу 50

3.3 Розробка клієнтської частини 55

4 ТЕСТУВАННЯ СИСТЕМИ 59

4.1 Unit тестування 59

4.2 Розробка інструкції користувача 73

4.3 Висновки 82

5 ЕКОНОМІЧНА ЧАСТИНА 83

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки 83

5.2 Оцінювання рівня новизни розробки……..86

5.3 Розрахунок витрат на проведення науково-дослідної роботи……90

5.4 Висновки……..105

ВИСНОВКИ……106

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……107

ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ..... 110

ДОДАТОК Б. ІЛЮСТРАТИВНИЙ МАТЕРІАЛ 113

ДОДАТОК В. ЛІСТИНГ МУТАЦІЇ ДЛЯ СТВОРЕННЯ IVR..... 121

ДОДАТОК Г. ЛІСТИНГ СЕРВІСУ РОБОТИ З IVR 123

ДОДАТОК Д. ЛІСТИНГ МУТАЦІЇ СТВОРЕННЯ НОВОГО СЦЕНАРІЮ IVR 134

ДОДАТОК Е. ЛІСТИНГ СЕРВІСУ ДЛЯ РОБОТИ ЗІ СЦЕНАРІЄМ IVR ……135

ДОДАТОК :.ЛІСТИНГ КОДУ АВТЕНТИФІКАЦІЇ..... 142

ДОДАТОК К.ЛІСТИНГ МОДУЛЯ ТЕСТУВАННЯ..... 144

ВСТУП

Обґрунтування вибору теми дослідження.

Природно, що бажання кожного абонента - клієнта - швидко отримати ясну відповідь на своє питання. Застосування системи IVR дозволяє зменшити час очікування в черзі на обслуговування. Дослідження показують, що близько 40% питань клієнтів досить прості і можуть бути легко автоматизовані, при цьому оператори зможуть зосередитися на обслуговуванні більш складних клієнтських запитів. В результаті автоматизації відповідей на прості запитання використання системи дозволяє скоротити час очікування з'єднання з оператором і зменшити число абонентів, які не дочекалися відповіді.

Використання системи IVR дозволяє цілодобово обслуговувати клієнтів без залучення для цієї мети додаткового персоналу. Крім того, система IVR не схильна до "зміни настрою", її "не дратує" настирливість клієнтів. Також відмічено, що деякі клієнти в силу своєї сором'язливості воліють спілкуватися з автоматичною системою, а не з оператором.

При побудові IVR можуть виникати помилки, які викликають невдоволення клієнтів, яке в багатьох випадках виправдане так, як неоптимально встановлення голосове меню вимагай багато часу і терпіння клієнта.

Для отримання максимальної ефективності від впровадження системи IVR, потрібно провести аналіз функцій автоматизації і методів їх виконання системою запитів IVR. Основна проблема IVR – оптимізація завантаження ресурсів та швидкого з’єднання клієнта з відповідальним за певну проблему менеджером

Тому актуальними є питання підвищення продуктивності та ефективності побудови IVR систем, оскільки існуючі методи не задовольняють потреби обслуговувати клієнтів.

Мета та завдання дослідження. Метою роботи є підвищення ефективності створення інтерактивного голосового меню за рахунок оптимізації передачі даних і використання системних ресурсів.

Основними задачами дослідження є:

- провести аналіз існуючих методів і засобів створення інтерактивного голосового меню;
- запропонувати нові:
- метод передачі даних за допомогою GraphQL для оптимізацію групової взаємодії користувачів;
- метод інтерактивного управління голосовим меню IVR, для оптимізації використання системних ресурсів для взаємодії з БД;
- розробити програмні компоненти та систему створення інтерактивного голосового меню;
- провести експериментальні дослідження розроблених засобів створення інтерактивного голосового меню.

Об’єкт дослідження – процес оптимізації створення інтерактивного меню IVR.

Предмет дослідження – методи та засоби створення інтерактивного меню IVR.

Методи дослідження. У процесі досліджень використовувались: методи теорії алгоритмів для розробки алгоритму і програмних модулів, методи контейнеризації, віртуалізації для створення dev-оточення, методи та способи оптимізації даних, методи теорії баз даних для розробки структури бази даних; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна одержаних результатів.

- Запропоновано метод передачі даних за допомогою GraphQL для оптимізації групової взаємодії користувачів, що дозволяє вилучити можливість помилки користувача.
- Подальшого розвитку отримав метод інтерактивного управління голосовим меню IVR, що дозволило оптимізувати використання системних ресурсів для взаємодії з БД.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській

кваліфікаційній роботі теоретичних положень запропоновано метод передачі даних за допомогою GraphQL для оптимізацію групової взаємодії користувачів, метод інтерактивного управління голосовим меню IVR, що дозволило оптимізувати використання системних ресурсів для взаємодії з БД.

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

Особистий внесок здобувача. У роботі [1] автору належить розробка інтерактивного додатку**.** Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто.

Апробація матеріалів магістерської кваліфікаційної роботи здійснена на Всеукраїнській інтернет-конференції «Модернізація та сучасні українські та світові наукові дослідження», м. Львів.

Публікації. Лічман Едуард Русланович. Розробка інтерактивих додатків / Лічман Едуард Русланович, Рейда Олександр Миколайович // Всеукраїнська інтернет-конференція «Модернізація та сучасні українські та світові наукові дослідження», м. Львів (2020).

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п'яти розділів, висновків, списку літератури, що містить 32 найменувань, 4 додатки. Робота містить 46 ілюстрацій, 5 таблиць. У першому розділі виконано аналіз стану питання та постановку задач досліджень. У другому розділі було розроблено структуру системи, методи та алоритм створення інтерактивного меню IVR меню. У третьому розділі було здійснено розробку програмних засобів та компонентів системи. У четвертому розділі було проведено тестування роботи системи. У п'ятому розділі проведено економічне обґрунтування доцільності та економічної вигоди даного проекту. У додатках міститься технічне завдання на роботу, лістинг коду, літистинг коду модульних тестів та ілюстративний матеріал до захисту роботи.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану питання

Як швидко розвиваються технології, так само швидко розвиваються потреби сучасного споживача. Слід прикласти максимум зусиль, щоб отримати перевагу над конкурентами та, звісно, здобути прихильність якомога більшої кількості клієнтів. Потік інформації у наш час є надзвичайно об’ємним, то ж потрібно ставити перед компанією конкретизовану та визначену ціль – привернути увагу ймовірного клієнта ще на стадії знайомства, ваша пропозиція має бути привабливішою та цікавішою для нього серед безліч інших аналогічних.

Беззаперечним є той факт, що одним із методів досягнення першості на ринку є використання сучасних технологій, а саме створення голосового меню IVR. Як правило IVR-запис містить основну інформацію про компанію та привітання. Використання цієї технології оптимізує роботу працівників, а саме: дозволить перенаправляти запити та нотувати інформацію в базу про клієнта автоматично, не залучаючи при цьому менеджера або ж секретаря. Така технологія є надзвичайно зручною та практичною, оскільки дозволить не лише зменшити обсяг робочих місць, а й структурує роботу компанії, налаштовує клієнта на подальшу співпрацю.

Усі сучасні компанії прагнуть максимально заощадити. Цей факт є очевидним, оскільки усі хотять отримати максимальний результат та, звісно, заробіток, затрачуючи мінімальну кількість ресурсів. Що дозволяє отримати компанії запис IVR меню?

Інтерактивна голосова відповідь відноситься до автоматизованої телефонної системи, яка дозволяє абонентам, що викликають, взаємодіяти з комп'ютером за допомогою попередньо записаних голосових повідомлень, перш ніж розмовляти з агентом-людиною.

Наприклад, якщо ви телефонуєте до комунальної компанії, заздалегідь записаний голос може вітати вас і дати вам кілька варіантів меню на вибір, таких як перевірка вашого рахунку, повідомлення про збій або зв'язок із представником служби підтримки клієнтів. Ви можете вибрати варіант усно або з телефонної клавіатури.

Якщо ви виберете такий варіант, як перевірка рахунку, IVR запросить інформацію про ваш лічильник або номер облікового запису, отримає доступ до інформації вашого облікового запису, а потім зачитає ваш рахунок або надішле його на ваш номер телефону.

Якщо ви хочете поговорити з представником служби підтримки клієнтів, IVR запросить інформацію про вашу проблему, а потім надішле ваш дзвінок будь-кому з відповідного відділу.

Іноді IVR навіть вирішує проблему клієнта, не передаючи її агенту-людині.

Сьогодні системи IVR досить продвинуті і можуть використовуватися для обробки багатьох типів запитів без необхідності участі людини – від бронювання рейсів та бронювання готелів до перевірки залишку на банківських рахунках, перевірки рахунків та поповнення рецептів на ліки.

Замовник отримує змогу створити нові реєстри своїх клієнтів, що значно облегшить роботу, адже інформацію можна буде подавати лиш зацікавленим у ній людям, а це у свою чергу дозволить значно підвищити об'єм успішних операцій. Представлена технологія здатна фіксувати реакції, тобто відстортувати клієнтів на тих, кому товар потрібен у найкоротші терміни, планують скористатися представленим товаром або послугою та тих, хто зовсім не зацікавлений у викладеній інформації.

Такі операції дозволять розробити власну статистику, яка допомагатиме у проведенні аналізу зібраної інформації.

Наступним пунктом із визначених переваг застосування IVR технологій буде економія. Це забезпечує максимальну мінімізацію витрачених ресурсів для досягнення поставленої мети. Слід памятати не тільки про економію коштів, але найголовніша перевага використання технології IVR– це збереження часу. Усім

відомо, що найціннішим ресурсом є час, і важко буде не погодитись, що усі сучасні розробки націлені саме на його збереження. То ж необхідно якомога глибше запроваджувати представлену технологію заради мінімізації витрачених ресурсів, що допоможе завоювати першість серед конкурентів.

Не слід забувати про надзвичайно важливий у наш час комп'ютеризації пункт – захист інформації. Використання IVR технологій дозволяє контролювати потоки інформації, відслідковувати усі клієнтські бази та рухи між ними. Люди належним чином ще недооцінюють важливість захисту інформації, адже належним чином виконана реалізація такого завдання є фундаментом успіху компанії.

Отже, у цій частині було розлянуто не лише сферу застосування та сучасний стан IVR технологій, а й переваги та конкретні сфери їх застосування, що робить розробку програмної системи оптимізації створення голосового меню IVR.

1.2 Порівняльний аналіз аналогів

На даний момент вибір платформи для створення IVR невеликий, проте є декілька, на перший погляд, зручних і функціональних веб-застосунків. Варто розглянути їх детальніше.

Найпопулярнішою платформою виявилась zadarma. Тут можна завантажити власні файли або просто надрукувати текст голосового меню і обраний користувачем голос озвучить написане. Також можна створювати шаблонні повідомлення. Проте застосунок дозволяє донести інформацію до клієнта, а не взаємодіяти з ним. Інтерфейс платформи наведено на рисунку 1.1.

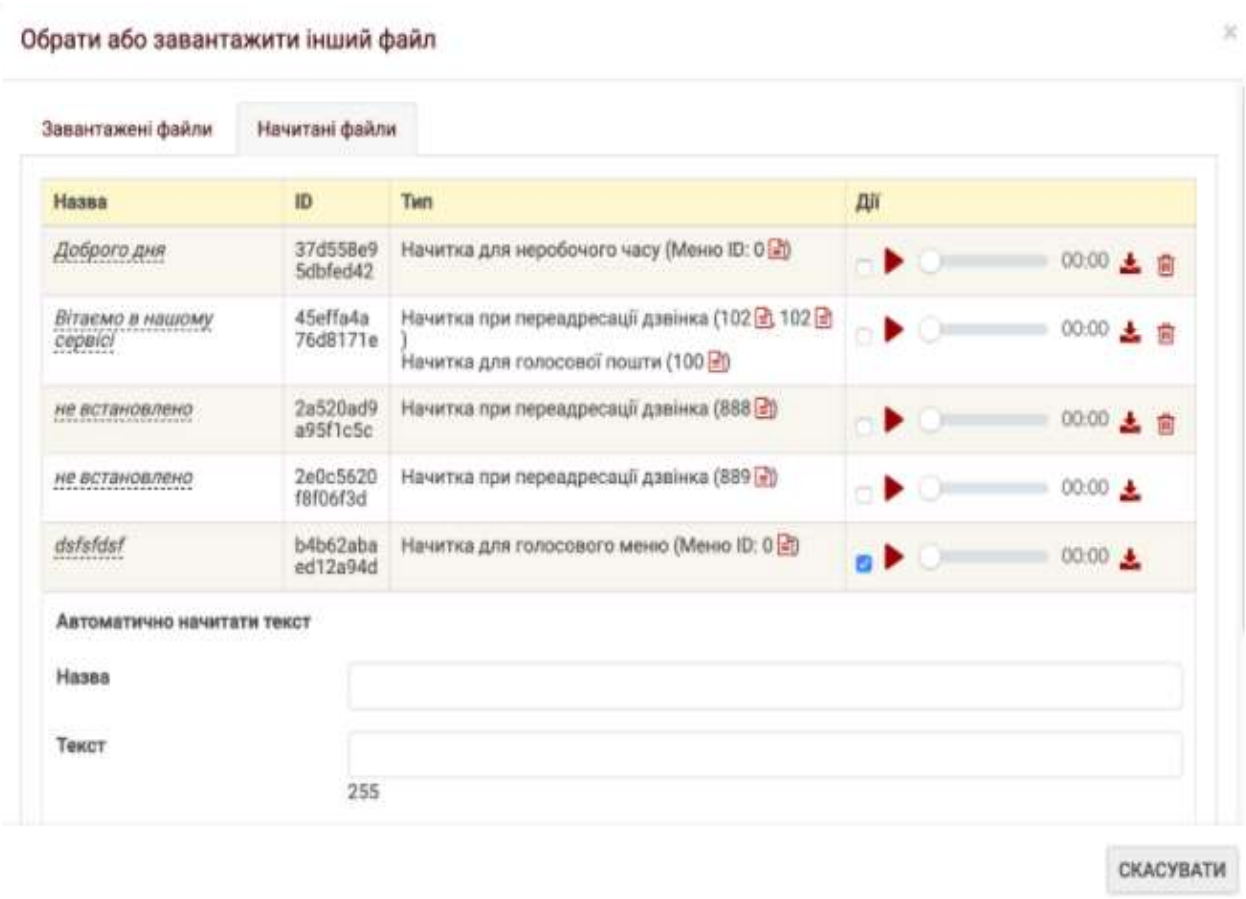

Рисунок 1.1 – Інтерфейс платформи zadarma

Також набирає популярності платформа aimylogic (рис. 1.2). На відміну від попередньої, на даній платформі користувач не може обрати голос озвучування чи записати розмову. Проте основний функціонал збережений : можна створити голосове меню зі своїх аудіо файлів, а для озвучування тексту можна обрати одну з 15 мов світу, яка записується її носієм, також є можливість взаємодіяти з клієнтом та обробляти отриману від нього інформацію, наприклад, відправляти запит з отриманими відомостями до Бази Даних, а потім повернути заздалегідь записане голосове повідомлення. Застосунок платний, оплачується похвилинно розмова.

Цікавим варіантом є платформа streamtele, зображена на рис. 1-3. Замовивши голосового помічника, користувач отримає можливість озвучити написаний текст будь-якими мовою та голосом, також можна взаємодіяти з клієнтами, записувати отриману від них інформацію та відстежувати дзвінки. Дані

можливості потрібно оплатити. Після огляду попередніх платформ, можна сказати, що дана має низку недоліків, наприклад, відсутня функція створення голосового меню з власних аудіо файлів або неможливість створення шаблонних повідомлень, що є дуже зручною та корисною функцією.

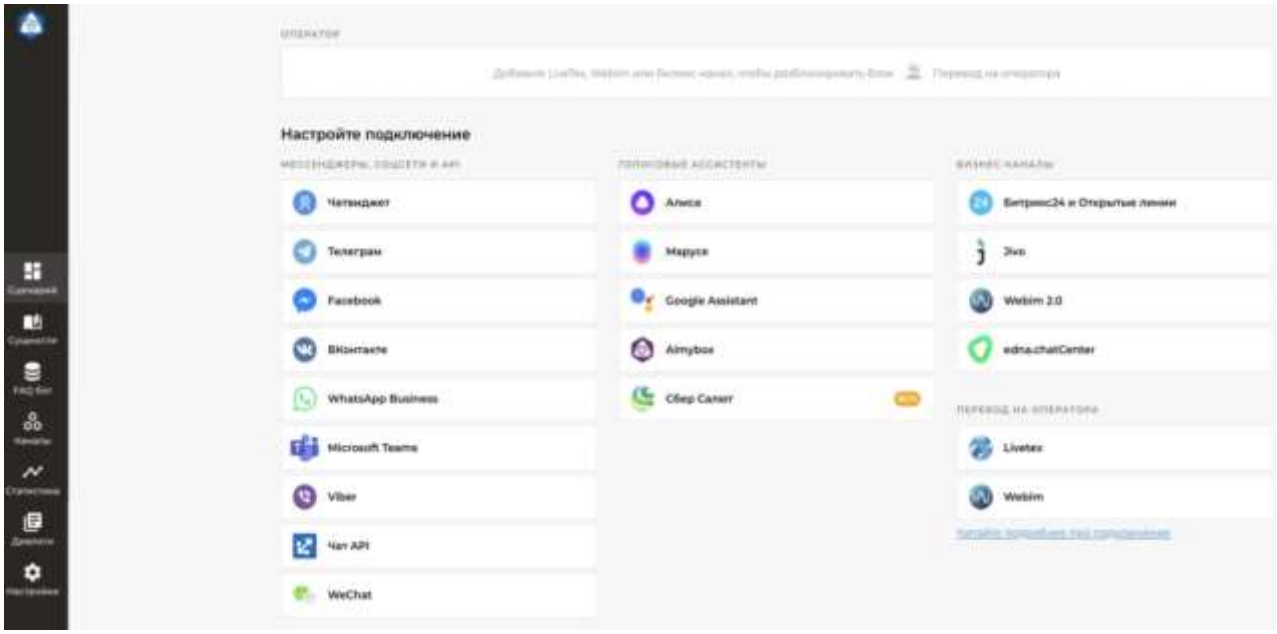

Рисунок 1.2 – Інтерфейс платформи aimylogic.com

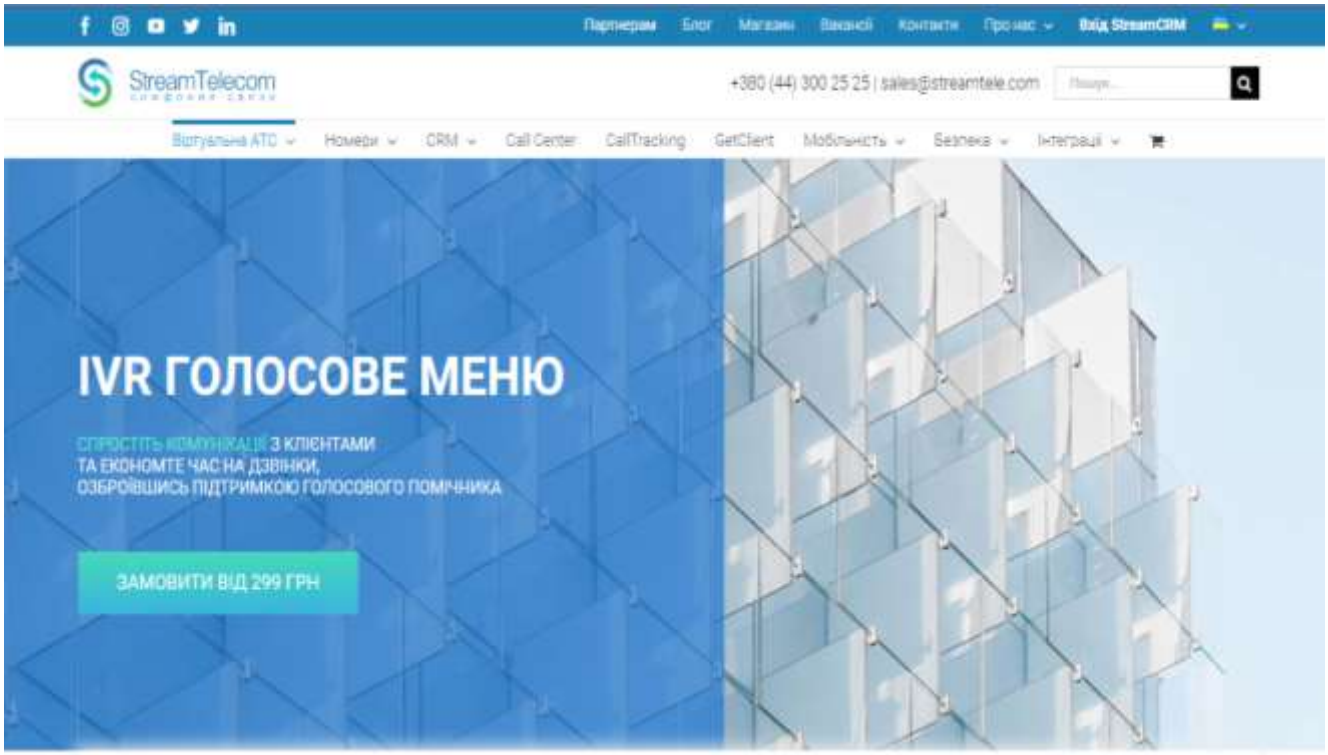

Рисунок 1.3 – Інтерфейс платформи streamtele.com

Порівняння власної розробки з аналогами наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика програмних продуктів

Критерій	zadarma.com	aimylogic.com	aimylogic.com	Project D (Власна розробка)
1	2	3	4	5
Створення голосового меню з власних аудіо файлів	1	1	0	1
Озвучування тексту на різних мовах	0	1	1	1
Вибір голосу озвучування	1	0	1	1
Обробка отриманої від абонента інформації	0	1	1	1
Безкоштовне використання	1	0	0	1
Запис розмови з абонентом	1	0	1	1
Створення шаблонних повідомлень	1	0	0	1
Сума	5	3	4	7

Таблиця порівняльних характеристик показала, що розробка власної системи є доцільною і в деякій мірі, необхідною, адже у даній галузі низька конкуренція. В результаті буде отримано продукт, який покриває недоліки існуючих рішень. Користувач буде мати доступ до всього необхідного функціоналу, а саме - створення голосового меню з власних аудіо файлів, дуже корисним буде можливість вибору голосу та мови озвучування, обробка отриманої від абонента інформації допоможе вести облік та статистику клієнтів,

задля безпеки та правильної обробки результатів, розмова буде записана, усі вищезгадані критерії користувач зможе отримати безкоштовно.

1.3 Аналіз алгоритмів створення голосового меню IVR

Серед алгоритмів по створенню голосового меню IVR виділяють два найпопулярніших: алгоритм створення однорівневого IVR-сценарію та створення багаторівневого IVR-сценарію [1].

Для того щоб створити однорівневий сценарій IVR необхідно виконати наступні кроки:

1. Ввести назву сценарію.
2. Якщо користувач встановлюватимете реакцію на голосові відповіді, то необхідно натиснути галочку «використовувати розпізнавання мови».
3. Вибрати дію, яку має зробити абонент і встановити на нього реакцію:
   - Записати відповідь – фіксування дії з боку абонента та завершення дзвінка.
   - Переадресація на номер – переадресація, також можна вказати для кожного абонента свій номер для переадресації.
   - Переадресація на SIP – переадресація на SIP телефонію, також можна вказати для кожного абонента свою sip-лінію для переадресації.
   - Надіслати смс клієнту – надсилання смс абоненту, також можна вказати для кожного абонента індивідуальний текст смс.
   - Надіслати смс замовнику – відправка смс із номером, що натиснув на вказаний номер, також можна вказати для кожного абонента індивідуальний номер замовника та текст смс.
   - Повтор запису натисканням – при натисканні необхідної цифри відбудеться повторення запису.
   - Додати до стоп-листу – додавання номера до спеціального телефонного списку (стоп-лист). На номери з даного списку дзвінки не здійснюватимуться.
   - Webhook – це повідомлення сторонніх додатків про події, що відбулися.

- Перейти до IVR # – перехід на вказаний ступінь у IVR-меню.
- Надіслати e-mail – надсилання електронного листа на вказану пошту з інформацією про натискання, також можна вказати для кожного номера телефону індивідуальний текст, електронну пошту, тему листа тощо.

4. Подія – це переадресація на номер або на SIP (тобто після закінчення аудиоролика відбувається переадресація, без будь-яких дій).
5. Додати ще IVR, якщо на якусь дію вказано реакцію «перехід до IVR».
6. Натиснути кнопку "Створити сценарій".

Наступний алгоритм - створення багаторівневого IVR-сценарію.

Якщо потрібно поставити абоненту кілька запитань і отримати відповіді, то можна налаштувати багаторівневий IVR-сценарій. Нижче по-кроково описано алгоритм створення сценарію з двома рівнями.

1. Ввести назву сценарію.
2. Якщо користувач встановлюватимете реакцію на голосові відповіді, то необхідно натиснути галочку «використовувати розпізнавання мови».
3. Вибрати дію, яку має зробити абонент і встановити на нього реакцію
   - Записати відповідь – фіксування дії з боку абонента та завершення дзвінка.
   - Переадресація на номер – переадресація, також можна вказати для кожного абонента свій номер для переадресації.
   - Переадресація на SIP – переадресація на SIP телефонію, також можна вказати для кожного абонента свою sip-лінію для переадресації.
   - Надіслати смс клієнту – надсилання смс абоненту, також можна вказати для кожного абонента індивідуальний текст смс.
   - Надіслати смс замовнику – відправка смс із номером, що натиснув на вказаний номер, також можна вказати для кожного абонента індивідуальний номер замовника та текст смс.
   - Повтор запису натисканням – при натисканні необхідної цифри відбудеться повторення запису.

- Додати до стоп-листу – додавання номера до спеціального телефонного списку (стоп-лист). На номери з даного списку дзвінки не здійснюватимуться.
- Webhook – це повідомлення сторонніх додатків про події, що відбулися.
- Перейти до IVR # – перехід на вказаний ступінь у IVR-меню.
- Надіслати e-mail – надсилання електронного листа на вказану пошту з інформацією про натискання, також можна вказати для кожного номера телефону індивідуальний текст, електронну пошту, тему листа тощо.

4. Далі у створенні сценарію встановлюємо реакцію на натискання 1 – перейти до додавання IVR - сценарію:
5. Обрати необхідний сценарій IVR або натиснути кнопку «Створити новий сценарій IVR»
6. У відповідному списку, що появиться необхідно перевірти правильність введених данних та натиснути кнопку «Наступний крок»
7. Обрати бажані аудиофайли, що будуть програватись при натисканні певної клавіші

Переваги такого підходу: систему можна адаптувати під більшість вимог клієнта: можна вказати бажані аудиофайли, які були записані попередньо на різних мовах, або ж записати потрібний прямо у вікні браузера, є можливість інтеграції у сторонні додатки, шляхом додання Webhook-сповіщень за потрібним url та за необхідних умов (коли відбувалась необхідна подія, наприклад, на голосовому меню IVR хтось перейшов у розід «Акції», бажає звязатися з оператором для проведення консультації, тощо), є можливість вибрати канал для сповіщення, через смс-повідомлення, через email або через сторонній сервіс, що є безперечно зручно та актуально у сучасно у бізнесі, можливість додання клієнтів до стоп-листу також є необхідною, адже це питання не лише стосується етики, а й інформаційної безпеки, неможливість заборонити користувачеві доступ до системи – не припустима [2]. Проте існують і недоліки: цей процес є досить не простим та без спеціальних знань іноді важко створити потрібний сценарій.

Враховуючи переваги та недоліки відомих методів реалізації створення голосового меню IVR, прийнято рішення адаптувати метод для можливості використання людям без спеціальних знань.

1.4 Постановка задачі розробки

Після проведення аналізу поточного стану та порівняння існуючих рішень за визначеними критеріями визначено завдання, які необхідно виконати, для розробки програмної системи оптимізації створення голосового меню IVR [3]:

1. Провести аналіз існуючих алгоритмів створення IVR.
2. Розробити алгоритм покращеного створення сценарію голосового меню IVR за рахунок усунення кроків, які можуть бути автоматизовані.
3. Розробити алгоритм роботи модуля, що відповідає за інтерактивний вибір стратегії гри.
4. Розробити API сервіс.
5. Розробити клієнтську частину.

Технічне завдання наведено у Додатку А.

1.5 Висновки

У першому розділі було розглянуто актуальність питання розробки IVR технологій, описано всі переваги використання, створення та область спеціалізації й застосування, визначено можливі шляхи покращення процесу створення голосового меню IVR та запропоновано доцільні шляхи вирішення недоліків. Оцінено реалізації присутніх на ринку програмних рішень та визначено недоліки (наприклад не було знайдено у конкурентів на ринку автоматизації більшості кроків, які можуть виконуватись без втручання користувача, які здатні зробити процес використання простішим та зрозумілішим для людини без знань вузької області роботи програмного забезпечення) у розробці та функціоналу, тому було прийнято рішення усунути проблеми у власній розробці. Було визначено основні

етапи розробки та перелік завдань до виконання – це дозволить структурувати процес розробки, що дозволить підвищити рівень ефективності роботи та, звісно, допоможе зберегти досить великий об’єм ресурсів, часу тощо. Було проаналізовано алгоритми створення створення голосового меню IVR, класифікацію та особливість роботи із кожним підвидом розробки. Детально розглянуто особливості основних конкурентів, які пропонують користувачам аналогічну роботу, що дозволило виокремити для своєї розробки деякі аспекти спеціалізації.

Отже, було проведено детальний аналіз, описано максимальну кількість переваг для створення власного продукту, що допоможе досягти першості серед конкурентів на ринку й втілити свої новаторські ідеї.

2 РОЗРОБКА СТРУКТУРИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз методів та засобів розробки веб-додатків

Клієнт-серверна архітектура набула своєї популярності завдяки динамічному розвитку мережі Інтернет та зосередження значної частини інформації в базах даних на серверах або у інших сервісах, до яких необхідно мати можливість звернутися для отримання даних. Її можна зазначити, як концепцію інформаційної мережі в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів. Така архітектура й визначає такі типи компонентів.

Правила взаємодії між клієнтом і сервером називаються протоколом обміну (протоколом взаємодії).

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

1. Рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
2. прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;
3. рівень управління даними, який забезпечує зберігання даних та доступ до них.

Сучасна програма, що розроблена у якості серверу надає API для доступу до сервісу, що його представляє. API має бути не змінним, зрозумілим і постійним, а не адаптований конкретно під одну реалізацію якогось особливого клієнта. API поинен бути доступний по HTTP (S) і забезпечувати доступ до всього функціоналу, який є в GUI або CLI.

Передача даних повина здійснюватись у максимально зрозумілому, легковісному форматі, доступним в загальноприйнятому, сумісному форматі,

такому як JSON. API надає об'єкти і служби в зрозумілій, організованій формі, наприклад, RESTful API або GraphQL забезпечують гідний інтерфейс [4].

GraphQL варто приділити особливу увагу тому що, технологія зявилась у 2012 році, проте була закритим проектом, до якого мав доступ лише одна компанія – Facebook. Показавши чудові результати, у 2015 році ця технологія була відкрита та доступна для всіх бажаючих, змогла за короткий час привернути увагу багатьох команд та крупних компаній.

GraphQL — це мова запитів для API та середовище виконання для виконання цих запитів із наявними даними. GraphQL надає повний і зрозумілий опис даних у вашому API, дає клієнтам можливість запитувати саме те, що їм потрібно, і нічого більше, полегшує розробку API з часом і забезпечує потужні інструменти розробника [5].

Надішліть запит GraphQL до свого API і отримайте саме те, що вам потрібно, ні більше, ні менше. Запити GraphQL завжди повертають передбачувані результати. Програми, які використовують GraphQL, швидкі та стабільні, оскільки вони контролюють дані, які вони отримують, а не сервер.

Запити GraphQL отримують доступ не тільки до властивостей одного ресурсу, але й плавно слідують посиланням між ними. У той час як типові API REST вимагають завантаження з кількох URL-адрес, API GraphQL отримують всі дані, необхідні вашій програмі, в одному запиті. Програми, які використовують GraphQL, можуть працювати швидко навіть при повільних підключеннях до мобільної мережі.

API GraphQL організовано за типами та полями, а не за кінцевими точками [6]. Ви маєте можливість отримати доступ до всіх можливостей ваших даних з однієї кінцевої точки. GraphQL використовує типи, щоб гарантувати, що програми запитують лише те, що можливо, і надають чіткі та зрозумілі помилки. Додатки можуть використовувати типи, щоб уникнути написання коду розбору (парсингу) вручну.

Є можливість додати нові поля та типи до свого API GraphQL, не впливаючи на наявні запити. Застарілі поля можуть бути застарілими та прихованими від

інструментів. Використовуючи єдину версію, що розвивається, API GraphQL надають програмам постійний доступ до нових функцій і заохочують до чистішого та зручнішого серверного коду.

GraphQL створює уніфікований API для всієї вашої програми, не обмежуючись певним механізмом зберігання. Написати API GraphQL, який використовують ваші наявні дані та код за допомогою механізмів GraphQL, доступних багатьма мовами. Ви надаєте функції для кожного поля в системі типів, і GraphQL викликає їх з оптимальним одночасністю.

Наступна технологія якій варто приділити увагу – ElastickSearch.

Elasticsearch – це одна з найпопулярніших пошукових систем у області Big Data, масштабоване нереляційне сховище даних з відкритим вихідним кодом, аналітична NoSQL-СУБД із широким набором повнотекстових функцій пошуку [7].

2.2 Розробка загальної моделі системи

Веб додаток представляє собою систему з клієнтською частиною та сервером, 6 додатками баз даних MySQL 8 версії, використанням Redis для зберігання даних у кеш. Загальна модель системи зображена на рисунку 2.1.

Elasticsearch – масштабована утиліта повнотекстового пошуку та аналітики, яка дозволяє швидко в режимі реального часу зберігати, шукати та аналізувати великі обсяги даних. ES є ядром ELK-стеку (Elastic Stack), до складу якого, крім Elasticsearch, входять такі продукти [8]:

1. Kibana
2. Logstash

Основні переваги та недоліки Elasticsearch розглянемо нижче. Варто зазначити, що одним із головних недоліків ES вважається схильність цієї NoSQL-СУБД до витоків даних через відсутність вбудованих засобів забезпечення інформаційної безпеки, таких як система авторизації та обмеження прав доступу.

Крім того, після установки двигун за замовчуванням зв'язується з портом 9200 на всі доступні інтерфейси, що відкриває доступ до бази даних.

ES забезпечує горизонтально масштабований пошук за допомогою багатопоточності. Система заснована на бібліотеці Apache Lucene, яка призначена для індексування та пошуку інформації у будь-якому типі документів. Всі функції Lucene доступні через API-інтерфейси на JSON та Java. ES дозволяє працювати з GET-запитами в реальному часі, але не підтримує розподілені транзакції. Безшовна інтеграція з Kibana гарантує легку керованість HTTP-інтерфейсом за допомогою JSON-запитів за рахунок REST API [9].

У масштабних Big Data системах кілька копій Elasticsearch об'єднуються у кластер. Пошукові індекси можна розділити на сегменти, реплікувавши кожен із яких кілька разів. Це забезпечує відмовостійкість системи. На вузлі ES кластера може розміщуватися кілька сегментів. Кожен вузол кластера діє як координатор для делегування операцій правильному сегменту з автоматичним перебалансуванням та маршрутизацією. Пов'язані дані часто зберігаються в тому самому індексі з одного або декількох первинних сегментів і декількох реплік. Після створення індексу кількість первинних сегментів не можна змінити. Довгострокове зберігання індексу забезпечує шлюз, дозволяючи відновлювати індекс при збої сервера.

Завдяки широкому набору функціональних можливостей, особливо повнотекстового пошуку за багатьма мовами та аналітикою в реальному часі, Elasticsearch активно застосовується в різних Big Data системах великих та середніх компаній по всьому світу. З найвідоміших зарубіжних користувачів варто відзначити корпорації Netflix, IBM, Facebook, Amazon, GitHub, Wikimedia, CERN, Mozilla, Adobe тощо. Так як даних очікується багато, нам необхідно обрати базу даних, яка б виконувала операцію пошуку найшвидше, а операції вставки чи видалення даних – посередньо. ES ідеально відповідає вимогам.

Docker – технологія контейнеризації, що дозволяє швидко розвернути проект з усіма залежностями на машині та робить розробку ефективною та передбачуваною [10].

Docker усуває повторювані повсякденні завдання конфігурації та використовується протягом усього життєвого циклу розробки для швидкої, легкої та портативної розробки додатків — настільних і хмарних. Комплексна наскрізна платформа Docker включає інтерфейси інтерфейсу користувача, інтерфейси командної команди, API та безпеку, які розроблені для спільної роботи протягом усього життєвого циклу доставки програми.

Контейнер — це стандартна одиниця програмного забезпечення, яка упаковує код і всі його залежності, щоб програма швидко й надійно працювала від одного обчислювального середовища до іншого [11]. Образ контейнера Docker — це легкий, окремий виконуваний пакет програмного забезпечення, який включає все необхідне для запуску програми: код, час виконання, системні інструменти, системні бібліотеки та налаштування.

Образи контейнерів стають контейнерами під час виконання, а у випадку контейнерів Docker – контейнери стають контейнерами, коли вони запускаються на Docker Engine. Доступне для програм на базі Linux і Windows, контейнерне програмне забезпечення завжди працюватиме однаково, незалежно від інфраструктури. Контейнери ізолюють програмне забезпечення від середовища і гарантують, що воно працює рівномірно, незважаючи на відмінності, наприклад, між розробкою та стадією.

Технологія контейнерів Docker була запущена в 2013 році як Docker Engine з відкритим вихідним кодом.

Технологія Docker унікальна, оскільки зосереджена на вимогах розробників і системних операторів щодо відокремлення залежностей додатків від інфраструктури.

Технологія, доступна від Docker і його проекту з відкритим кодом, Moby була використана всіма основними постачальниками центрів обробки даних і хмарними провайдерами. Багато з цих постачальників використовують Docker для своїх контейнерних пропозицій IaaS. Крім того, провідні безсерверні фреймворки з відкритим вихідним кодом використовують технологію контейнерів Docker.

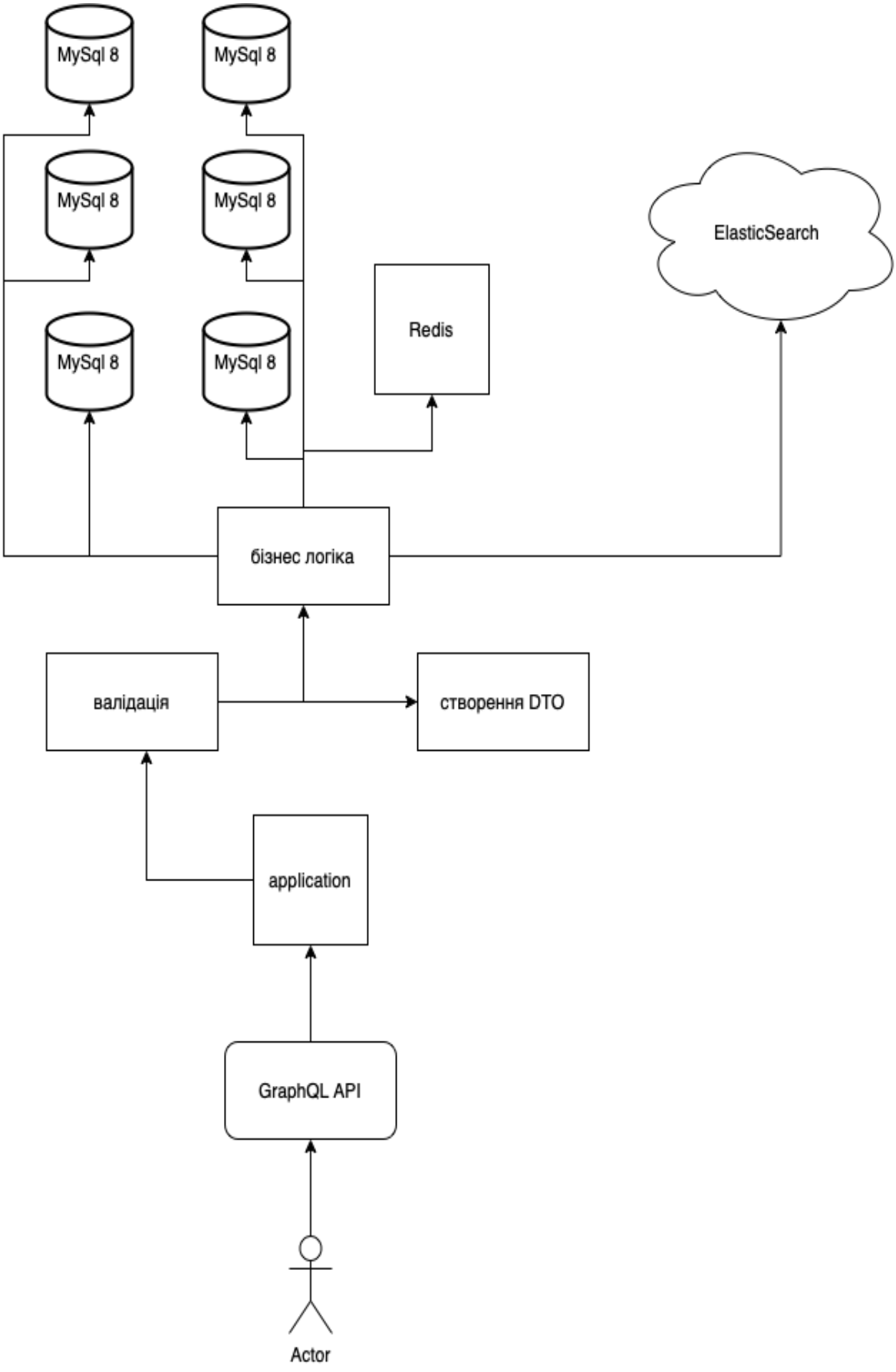

Рисунок 2.1 – Загальна модель додатку

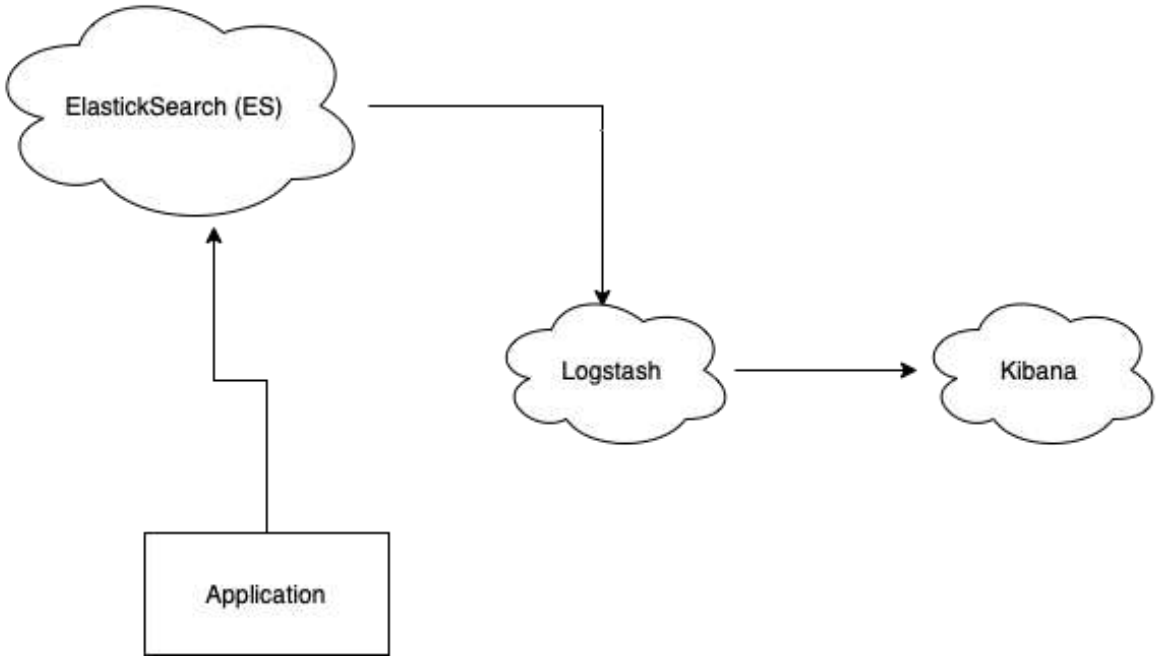

Рисунок 2.2 - Схема взаємодії ELK стеку

Існує щонайменше два основних підходи до проектування клієнт серверних програмних систем: монолітний та мікросервісний. Кожен з цих підходів має свої переваги та недоліки.

Монолітна архітектура полягає у використанні одного застосунку, який включає в себе всі необхідні функції. Перевага даного підходу: простота реалізації, швидкість розробки першої версії додатку, простота розгортання. Недоліки такого підходу: з ростом бази коду, застосунок стає все складніше підтримувати та додавати нові функції, оскільки існує сильний взаємозв'язок між різними компонентами системи; погана маштабованість, оскільки доступне тільки вертикальне маштабування, без можливості горизонтального маштабування [12].

Іншим підходом – є використання мікросервісної архітектури, яка вирішує більшість проблем монолітного підходу. Перевагами такого підходу є те, що різні компоненти системи ізольовані між собою, що не створює зайвих залежностей, дозволяє спростити розробку нового функціоналу, а також тестованість та підтримку існуючого. Іншою перевагою є підвищення маштабованості, оскільки у разі високого навантаження на сервіс є можливість розгорнути нові екземпляри контейнера та сховища даних, задля горизонтального маштабування. Недоліком

такого підходу в порівнянні з монолітним є необхідність розробки додаткового функціоналу для комунікації між сервісами, що не є проблемою враховуючи усі переваги названі вищу.

Виходячи з вимог до системи було прийнято рішення застосувати мікросервісну архітектуру [13]. Архітектура системи наведена на рисунку 2.3.

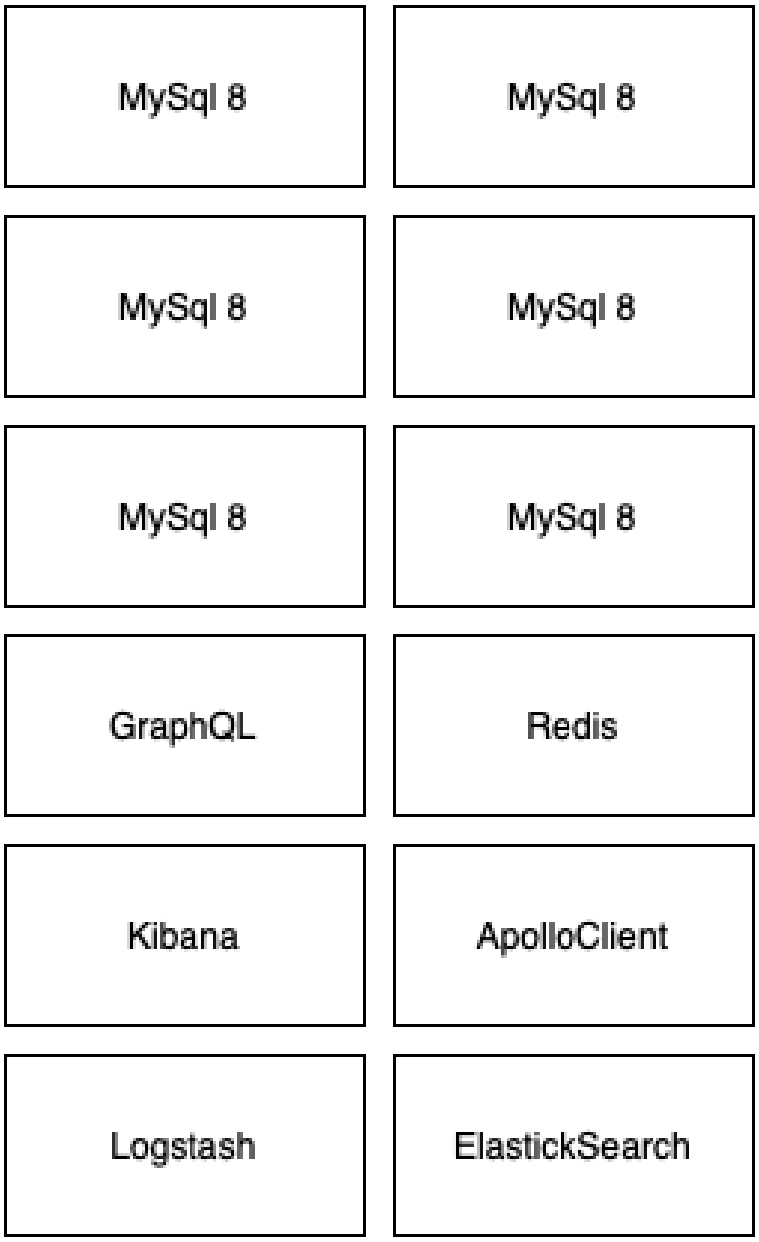

Рисунок 2.3 – Архітктура системи

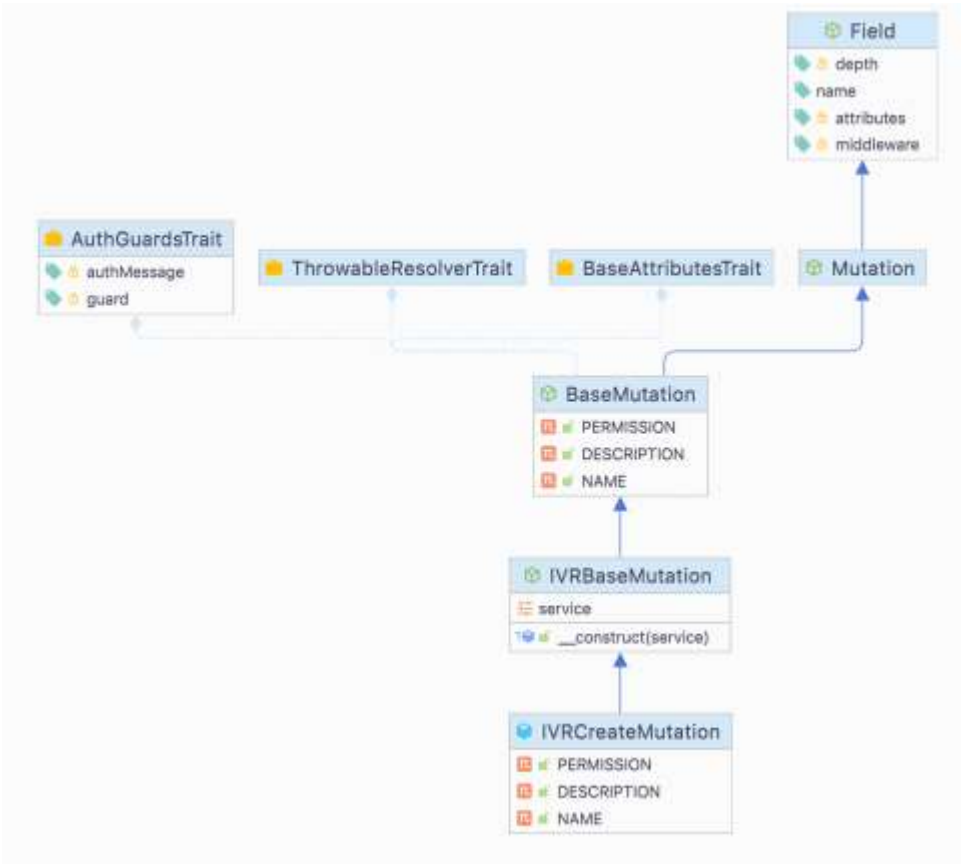

Рисунок 2.4 – Діаграма класів модуля створення IVR

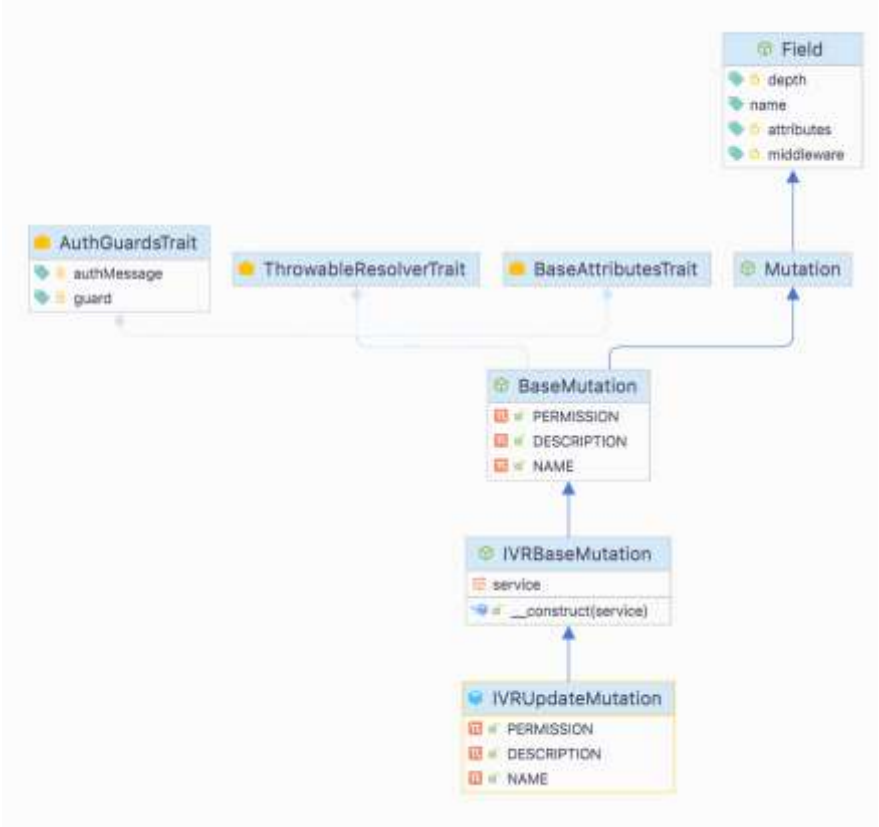

Рисунок 2.5 – Діаграма класів модуля оновлення IVR

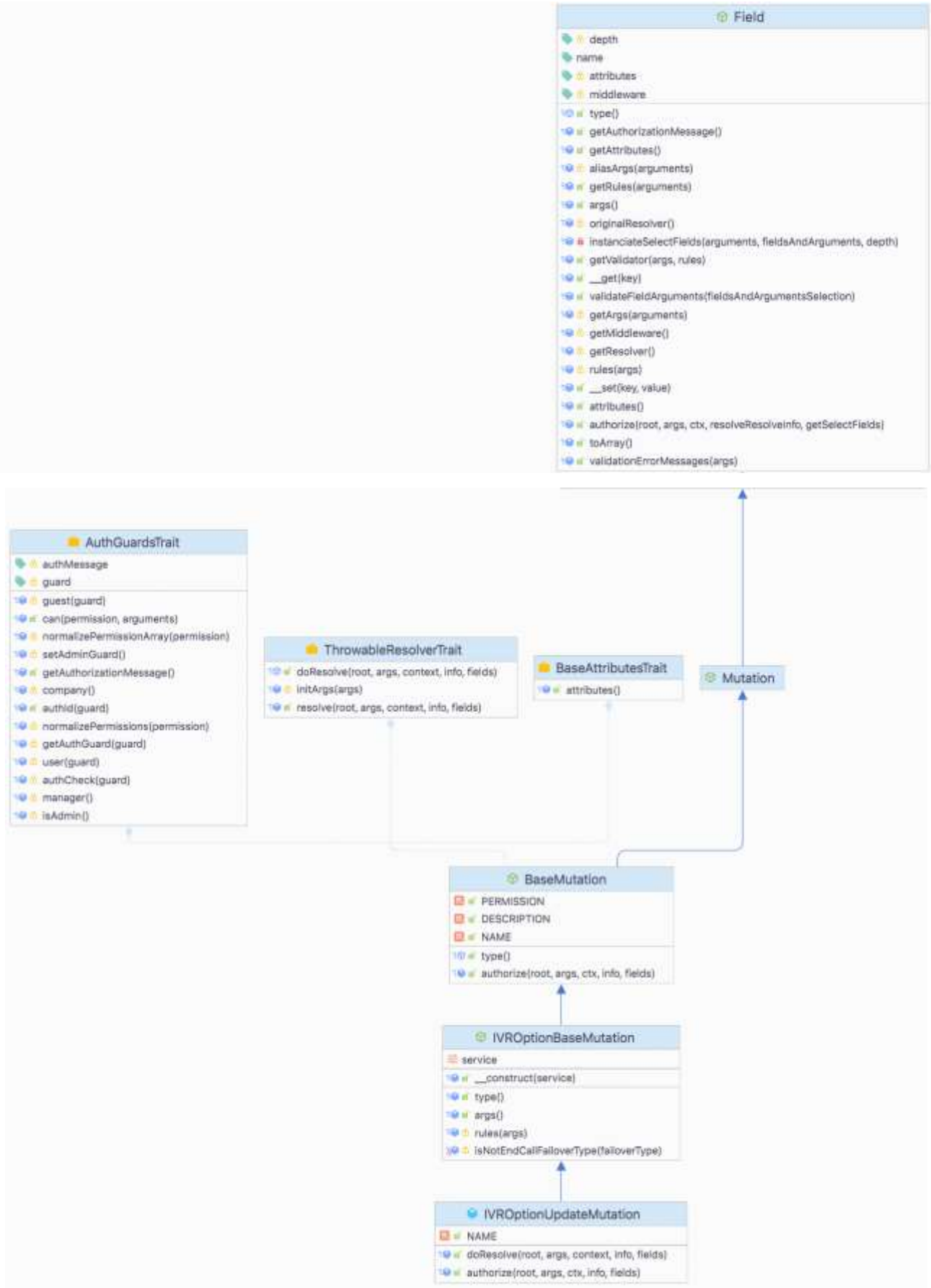

Рисунок 2.6 – Діаграма класів модуля оновлення сценарію IVR

```
el:
  image: docker.elastic.co/elasticsearch/elasticsearch-oss:7.9.3
  container_name: ${APP_NAME}_elastic
  hostname: ${APP_NAME}_elastic
  environment:
    - discovery.type=single-node
    - node.name=${APP_NAME}_elastic
    - cluster.name=ovpdev-es-cluster
  restart: unless-stopped
  ports:
    - 9200:9200
    - 9300:9300
  depends_on:
    - php

kibana:
  container_name: ${APP_NAME}_kibana
  hostname: ${APP_NAME}_kibana
  image: docker.elastic.co/kibana/kibana-oss:7.9.3
  environment:
    - ELASTICSEARCH_HOSTS=http://${APP_NAME}_elastic:9200
  ports:
    - 5601:5601
  depends_on:
    - el
```

Рисунок 2.7 - Приклад конфігурації docker-compose файлу для ES сервісу

```
db_radius:
  image: mariadb:10.3
  container_name: ${APP_NAME}_db_radius
  volumes:
    - ./database/dumps/radius:/docker-entrypoint-initdb.d:ro
  ports:
    - 3310:${DB_PORT}
  environment:
    MYSQL_ROOT_PASSWORD: ${DB_RADIUS_PASSWORD}
    MYSQL_DATABASE: ${DB_RADIUS_DATABASE}

cache:
  image: nbtri/alpine-redis
  environment:
    REDIS_PASSWORD: ${REDIS_PASSWORD}
  container_name: ${APP_NAME}_cache
  hostname: ${APP_NAME}_cache
  ports:
    - ${REDIS_PORT}:${REDIS_PORT}
```

Рисунок 2.8 – Приклад конфігурації docker-compose файлу для сервісу кеша та бази даних

2.3 Розробка діаграми послідовності

Діаграма послідовності – відображає впорядковану за часом взаємодію об’єктів, тобто відображають об’єкти, які задіяні та послідовність відправлення і отримання повідомлень.

Пропонується в діаграмі послідовності розглянути основний сценарій використання такий, як логін в систему. Цей сценарії проілюстровано на рисунку 2.9.

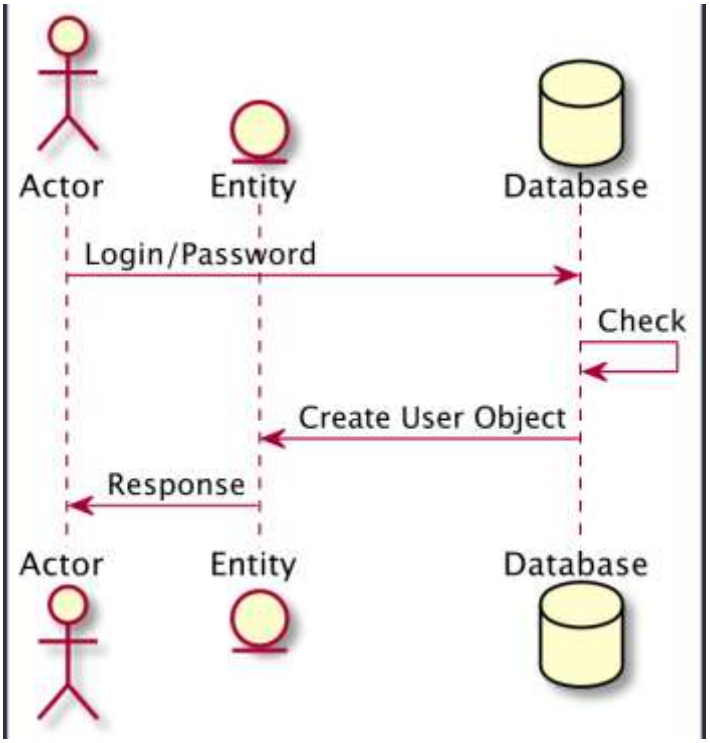

Рисунок 2.9 – Діаграма послідовності логіну

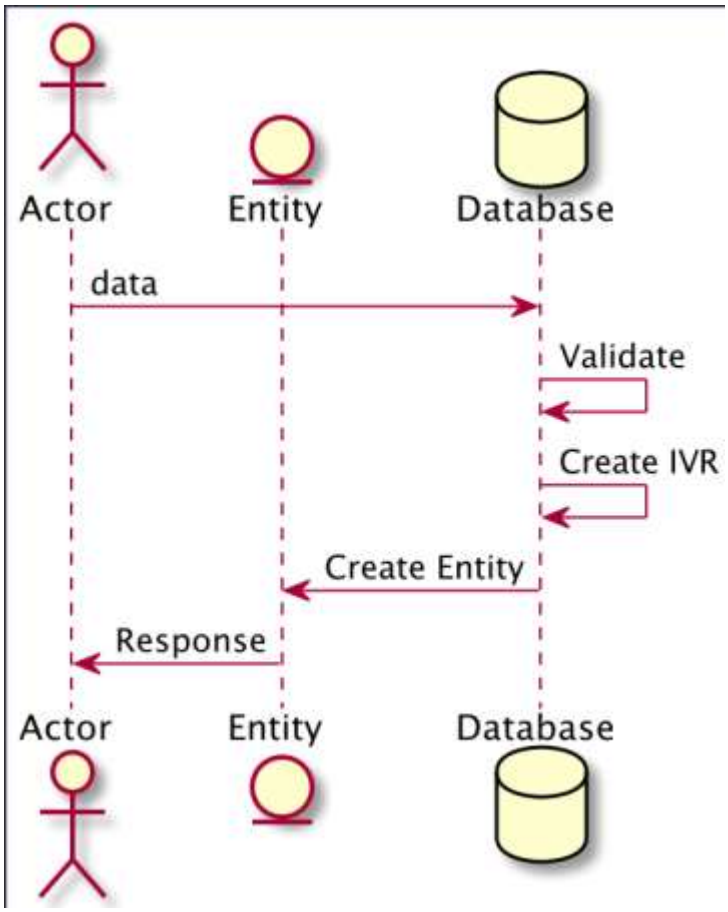

Рисунок 2.10 – Діаграма послідовності створення IVR

2.4 Розробка алгоритму створення голосового меню IVR

Голосове меню (IVR) дозволяє управляти рухом виклику за допомогою DTMF-посилок натисканням відповідних клавіш-пунктів меню, і в результаті направляти користувача на внутрішній номер, додаток, набирати внутрішній номер в явному вигляді, програвати відповідні вітання [14]. Список IVR платформи зберігається в таблиці ivrs БД dbasterisk_cfg.

Структура таблиці наступна:

Таблиця 2.1 – Структура таблиці ivrs

Назва поля	Тип даних	Опис
1	2	3
id	bigint(11) unsigned	ІД запису
domain_id	int(10) unsigned	ІД домену, в якому створюється IVR
ivr_number	varchar(50)	Номер IVR
announcement	varchar(60)	ім'я запису привітання, що програється при вході в IVR
timeout_failover	varchar(50)	внутрішній номер для переходу по закінченню очікування введення опції меню
timeout_announcement	varchar(60)	ім'я запису привітання, що програється по закінченню очікування введення опції меню
invalid_announcement	varchar(60)	ім'я запису привітання, що програється якщо користувач вибрав недоступну опцію меню
attempts	int(11)	кількість допустимих невірних спроб введення опції меню поспіль, після чого користувач буде виведений з меню
attempts_failover	varchar(50)	внутрішній номер для переходу по закінченню допустимих неправильних спроб введення опції меню поспіль

Продовження табл. 2.1

1	2	3
attempts_announcement	varchar(60)	ім'я запису привітання, що програється якщо користувач вичерпав число допустимих неправильних спроб введення опції меню
hours	varchar(80)	діапазон годин і хвилин доби протягом яких даний IVR Активний. Допустимі значення: 00:00-13:00 для вказівки діапазону, або * - для вибору будь-якого часу доби
weekdays	varchar(80)	діапазон днів тижня протягом яких даний IVR Активний. Допустимі значення: sun mon tue wed thu fri sat для вказівки одного дня, або для вказівки діапазону днів sun-mon або * - для вибору будь-якого дня тижня
mdays	varchar(80)	діапазон днів місяця протягом яких даний IVR Активний. Допустимі значення: 1 2 3 4 … 31 для вказівки одного дня, або для вказівки діапазону днів: 1-10 або * для вказівки будь-якого дня

Продовження табл. 2.1

1	2	3
month	varchar(80)	діапазон місяців у році протягом яких даний IVR активний. Допустимі значення: jan feb mar apr may jun jul aug sep oct nov dec для вказівки конкретного місяця, або для вказівки діапазону місяців: may-nov , або * для вказівки будь-якого місяця в році
schedule_failover	varchar(60)	внутрішній номер для переходу в разі якщо виклик надійшов в IVR поза часом активності
timeout	int(10)	число в секундах протягом якого очікується введення від користувача в меню IVR
return_code	varchar(1)	опція повернення на попередній рівень меню. Допустимі значення:*, 0-9. Символ # неприпустимий, будь-які інші символи неприпустимі.
record	tinyint(1)	прапорець включення запису викликів знаходяться в IVR. 1-якщо запис увімкнено, NULL - якщо вимкнено.
extdial_announcement	varchar(60)	ім'я запису привітання, що програється в якості запрошення для пунктів меню, що виконують набір внутрішнього номера

Продовження табл. 2.1

1	2	3
extdial_invalid_announcement	varchar(60)	ім'я запису привітання, що програється в якості повідомлення про неприпустиме введення для пунктів меню, що виконують набір внутрішнього номера
language_default	varchar(10)	код мови встановлюється за замовчуванням для даного IVR
language_change	tinyint(1)	прапорець дозволяє або забороняє викликати меню вибору мови до початку обробки пунктів основного меню IVR. 1-дозволяє, NULL-забороняє
language_announcement	varchar(255)	ім'я запису привітання, що програється в якості запрошення вибрати мову у відповідному меню

Список доступних для вибору мов локалізації IVR перераховується в таблиці ivr_languages.

Таблиця 2.2 – Структура таблиці ivr_languages:

Назва поля	Тип колонки	Опис
id	bigint(11) unsigned	ІД запису
domain_id	int(10)	ІД домену в якому знаходиться IVR, для якого закріплюється опція мови
ivr_number	varchar(50)	Номер IVR (без частини із зазначенням варіативності) для якого закріплюється опція мови
option	int(10)	Значення клавіші, яка відповідає вибору даної мови.
language	varchar(10)	Код мови, який буде встановлений для голосового меню, при виборі даної опції

Для того, щоб додати нове меню IVR потрібно виконати наступні кроки: Додати новий запис про голосове меню в таблицю ivrs БД dbasterisk_cfg:

1. id-ІД запису, буде згенерований автоматично при додаванні

2. domain_id - ІД домену, якому належить доданий IVR
3. ivr_number - внутрішній номер голосового меню. Дане значення формується наступним чином: задається базовий внутрішній номер даного меню, наприклад 1010. За цим номером дане меню доступне в системі телефонії. Після цього в залежності від того, чи є це першою варіацією даного меню, до нього додається через роздільник номер варіації. Варіації голосових меню з'являються як наслідок створення різних розкладів активності одного і того ж голосового меню. Якщо на момент створення меню воно є першим з таким номером, через роздільник (крапка) вказується значення 1. Таким чином отримуємо кінцеве значення для даного поля як 1010.1.
4. Якщо в подальшому буде доданий варіант даного голосового меню з відмінними налаштуваннями розкладу, необхідно використовувати інший номер варіації, притому не обов'язково зберігати послідовність.. Допустимими варіантами можуть бути-1010.2, 1010.5, 1010.6, і тд.
5. announcement - ім'я файлу привітання, яке буде програно при вході в IVR. Наприклад-welcome [15]
6. timeout_failover-внутрішній номер користувача або програми для переходу по закінченню таймауту очікування введення опції меню від користувача. Якщо вказано номер, система буде намагатися з'єднати з ним користувача, що знаходиться в меню, наприклад - 10001, якщо значення поля NULL - відбудеться роз'єднання.
7. timeout_announcement - ім'я запису привітання, що програється по закінченню очікування введення опції меню. Наприклад-bye.
8. invalid_announcement - ім'я запису привітання, що програється якщо користувач вибрав недоступну опцію меню. Наприклад-invalid.
9. attempts-кількість допустимих неправильних спроб введення опції меню поспіль, після чого користувач буде виведений з меню на напрямок . Наприклад-3.

10. attempts_failover-внутрішній номер для переходу по закінченню допустимих неправильних спроб введення опції меню поспіль. Якщо вказано номер, система буде намагатися з'єднати з ним користувача, що знаходиться в меню, наприклад - 10001, якщо значення поля NULL - відбудеться роз'єднання.
11. attempts_announcement - ім'я запису привітання, що програється якщо користувач вичерпав число допустимих неправильних спроб введення опції меню поспіль. Наприклад-wronginput.
12. hours-години в які активно дане голосове меню. Допустимі значення цього поля наведені в таблиці вище. Наприклад-10:00-17:00
13. weekdays-дні тижня, в які активне дане голосове меню. Допустимі значення цього поля наведені в таблиці вище. Наприклад-mon-fri
14. mdays-дні місяця, в які активне дане голосове меню. Допустимі значення цього поля наведені в таблиці вище. Наприклад - *
15. month-місяці в році, в які активне дане голосове меню. Допустимі значення цього поля наведені в таблиці вище. Наприклад - *
16. schedule_failover-внутрішній номер для переходу в разі якщо виклик надійшов в IVR поза часом активності. Якщо вказано номер, система буде намагатися з'єднати з ним користувача, що знаходиться в меню, наприклад - 10001, якщо значення поля NULL - відбудеться роз'єднання.
17. timeout-число в секундах протягом якого очікується введення від користувача в меню IVR. Наприклад-30
18. return_code-клавіша, яка буде використана в якості опції повернення на рівень вище. Допустимі значення наведені в таблиці вище. Наприклад - *
19. record-прапорець включення запису викликів знаходяться в IVR. 1-якщо запис увімкнено, NULL - якщо вимкнено. Наприклад-1
20. extdial_announcement - ім'я запису привітання, що програється в якості запрошення для пунктів меню, що виконують набір внутрішнього номера. Наприклад-letthepartybegin

21. extdial_invalid_announcement - ім'я запису привітання, що програється в якості повідомлення про неприпустиме введення для пунктів меню, що виконують набір внутрішнього номера. Наприклад-wrongnumber
22. language_default-код мови встановлюється за замовчуванням для даного IVR. Наприклад-ru_3 якщо IVR належить компанії з ІД-3.
23. language_change-прапорець дозволяє або забороняє викликати меню вибору мови до початку обробки пунктів основного меню IVR. 1-дозволяє, NULL-забороняє. Наприклад-1.
24. language_announcement - ім'я запису привітання, що програється в якості запрошення вибрати мову у відповідному меню вибору мови.

Внести запис в таблицю ivr_languages БД DB asterisk_cfg. Цей пункт необхідно виконувати, у разі, якщо в п.1 для поля language_change було встановлено значення 1.

1. id-ІД запису, буде згенерований автоматично при додаванні
2. domain_id-вказати ІД домену з таблиці domain_id, отриманий при створенні домену, якому належить створюване голосове меню, для якого закріплюється опція вибору мови. Наприклад-1
3. err_number-вказати номер IVR з п.1 без частини із зазначенням варіативності для якого закріплюється опція вибору мови. Наприклад-1010
4. option-вказати значення опції, для якої закріплюється код вибраної мови. Це номер клавіші, при натисканні на яку, буде обрано ту чи іншу мову. Наприклад-1
5. language-вказати код мови, який буде встановлений, при виборі опції, заданої для поля option [16]. Наприклад-ru_1 якщо ми вказуємо мову з кодом ru а IVR належить компанії з ІД 1
6. 3. Внести запис в таблицю extensions_global БД dbasterisk_cfg:
7. Важливо: запис в дану таблицю вноситься тільки при створенні першої варіації голосового меню. Якщо додається додатковий варіант меню з

відмінним від основного меню розкладом-цей пункт виконувати не потрібно.

8. id-ІД запису, буде згенерований автоматично при додаванні
9. app_id-завжди NULL в нашому випадку
10. domain_id-вказати ІД домену з таблиці domain_id, отриманий при створенні домену, якому належить створюване голосове меню. Наприклад-1
11. extends_number-вказати сформоване значення поля ivr_number без частини з ідентифікатором варіації з (1). Наприклад 1010 для першої створеної варіації меню 1010.1.
12. extern_alias-вказати значення з exten_number. Наприклад-1010.
13. app_type_queue - в даному випадку-NULL
14. app_type_followgroup - в даному випадку-NULL
15. app_type_ivr - при додаванні запису про голосовому меню-завжди 1.
16. forward - в даному випадку NULL
17. Після цього запис про IVR меню доданий. Необхідно перейти до створення пунктів даного меню.

2.5 Висновки

В другому розділі проведено аналіз методів та засобів розробки веб додатків, було проаналізовано переваги різних технологій та їх недоліки, розроблено загальну модель системи, що враховує складність запуску у dev оточенні за рахунок необхідності одночасного доступу до 6 додатків баз даних MySQL 8, сервісу ElastickSearch, сервісу Kibana та сервісу Logstash та реалізовано оптимізований алгоритм створення голосового меню IVR [17].

Розроблено архітектуру системи, яка базується на мікросервісному підході. Розроблено діаграму послідовності для сценарію входу у систему.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ

3.1 Варіантний аналіз і обґрунтування вибору засобів реалізації програмного засобу

Нам необхідно переглянути перелік мов програмування, які можуть виконуватися на сервері. Такими мовами є: PHP, Java. Усі вони мають свої сильні та слабкі сторони, які і зумовлюють різні сфери їх використання.

Java – має універсальний функціонал то ж її область використання є надзвичайно великою. Java можна зустріти в усіх областях людського життя, завдячуючи її універсальності [18].

Java не можна назвати складною, то ж за її вивчення можуть сідати навіть новачки. Ця мова доступна на усіх платформах, хоча й весь час оновлюється, це не заважає старому коду працювати .

Існує багато програм від Java, від сайтів електронної комерції до програм для Android, від наукових до фінансових програм, від ігор, таких як Minecraft, до настільного програмного забезпечення, такого як Eclipse, Netbeans і IntelliJ, від фреймворків з відкритим кодом до програм J2ME. , і т. д. Розглянемо кожен з них докладніше.

Java надзвичайно широко використовується у фінансовій сфері. Багато глобальних інвестиційних банків, таких як Goldman Sachs, Citigroup, Barclays, Standard Charted та інші, використовують Java для написання зовнішніх і внутрішніх офісних електронних систем, систем регулювання та підтвердження, проектів обробки даних.

Java також широко застосовується в електронній комерції та веб-додатках. Величезна кількість сервісів RESTful було створено за допомогою Spring MVC, Struts 2.0 і подібних фреймворків. Навіть найпростіші програми на основі Servlet, JSP і Struts досить популярні в різних державних проектах. Багато веб-додатків уряду, охорони здоров'я, страхування, освіти, оборони та деяких інших відомств написані на Java.

Нижче, на рисунку 3.1 зображено приклад програмного коду на мові програмування Java.

AddElementsOfList.java × ModifyFunctionParameter.java × Customer.java ×

```
5   import java.util.*;
6   public class AddElementsOfList {
7
8       public static void main(String [] args){
9
10          //create a list of even numbers
11          List<Integer> mylist = new ArrayList<Integer>();
12          for(int i = 1; i < 5; i ++){
13              mylist.add(new Integer(i));
14          }
15
16          // adding elements in a list
17          int intresult = 0;
18          for(Integer n: mylist ){
19              intresult +=  n.intValue();
20          }
21          System.out.println(intresult);
22      }
23  }
24
```

Рисунок 3.1 – Приклад програмного коду на Java

Додатки для торгівлі сторонніми розробниками, які також є частиною великої індустрії фінансових послуг, також використовують Java. Популярні програми, такі як Murex, які використовуються в багатьох банках, написані на Java.

Хоча поява iOS і Android практично знищила ринок J2ME, все ще існує величезна кількість дешевих телефонів від Nokia і Samsung, які використовують J2ME. Були часи, коли майже всі ігри та програми, доступні на Android, були написані за допомогою MIDP і CLDC, які є частиною платформи J2ME. J2ME все ще популярний у таких медіа, як Blu-ray, карти та телеприставки. Однією з причин популярності WhatsApp є те, що він також доступний на J2ME [19].

Hadoop та інші технології великих даних так чи інакше використовують Java, наприклад Hbase і Accumulo від Apache або ElasticSearch. Хоча Java в цій сфері не домінує, оскільки існують такі технології, як MongoDB, які написані на

C++. Java може отримати велику частку цієї зростаючої області, якщо Hadoop або ElasticSearch розширяться [20].

Java покращила свою продуктивність і за допомогою сучасних JIT здатна забезпечити продуктивність на рівні C++. З цієї причини Java також популярна для написання високопродуктивних систем, тому що, хоча продуктивність поступається рідній мові, ви можете пожертвувати безпекою, мобільністю та надійністю заради більшої швидкості, і потрібен лише один недосвідчений програміст C++, щоб зробити програму повільною. і ненадійний. .

PHP розшифровується як «Hypertext Preprocessor». Синтаксис мови заснований на Perl, Java і C. PHP є досить простою мовою, її легко вивчити. Перевага PHP полягає в тому, що він дає веб-майстрам можливість створювати динамічні (генеровані) сторінки HTML [21].

Основна перевага PHP перед такими мовами, як C і Java, полягає в тому, що PHP можна використовувати для створення HTML-сторінок з PHP-командами, вбудованими в HTML-документ.

PHP-скрипти виконуються на стороні сервера, а не на стороні клієнта, наприклад JavaScript [22]. WEB-сервер можна налаштувати так, щоб процесор PHP обробляв сторінки HTML: користувачі можуть не знати, чи отримують вони статичні чи динамічні сторінки HTML.

Мова сценаріїв PHP дозволяє швидко розробляти якісні WEB-додатки, які можна буде підтримувати в майбутньому і при необхідності легко модифікувати.

PHP простий у освоєнні і в той же час потужний інструмент для професійних програмістів. Вивчити PHP не складно, навіть якщо ви чуєте про нього вперше в житті. Кілька годин, витрачених на вивчення PHP, буде достатньо для написання простих скриптів на PHP.

PHP домінує серед мов програмування для Інтернету і постійно вдосконалюється.

Мова сценаріїв PHP має великий потенціал. Основною сферою застосування PHP є створення скриптів, що виконуються сервером. Це дозволяє сценарію PHP виконувати те, що роблять програми CGI. Зокрема, PHP дозволяє обробляти дані

форми, отримувати та надсилати файли cookie, а також генерувати динамічні сторінки HTML. Крім того, мова сценаріїв PHP надає величезну кількість інших завдань WEB-програмування.

Нижче, на рисунку 3.2 зображено приклад програмного коду на PHP.

```
public function doResolve(
    $root,
    array $args,
    $context,
    ResolveInfo $info,
    SelectFields $fields
): bool {
    return $this->localeService->delete(
        Locale::query()->findOrFail($args['id'])
    );
}

protected function rules(array $args = []): array
{
    return [
        'id' => ['required', 'integer', 'exists:locales,id'],
    ];
}
```

Рисунок 3.2 – Приклад програмного коду на php

Три основні сфери застосування PHP.

1. Написання сценаріїв, які запускаються на стороні сервера. Це сфера застосування PHP. Для виконання завдання потрібен WEB-сервер, браузер і PHP-парсер, який може бути у вигляді серверного модуля або CGI-програм [23]. Для перегляду результатів виконання в браузері скриптів, написаних на PHP, потрібен WEB-сервер і встановлений на ньому PHP.

2. Написання скриптів, які можуть запускатися в командному рядку незалежно від WEB-сервера і навіть браузера. Для виконання цього завдання потрібен тільки парсер PHP. Такі скрипти широко використовуються для обробки текстів. Крім того, цей спосіб запускається під Windows або для регулярно створюваних скриптів, зокрема, за допомогою планувальника завдань, планувальника завдань, cron під Linux або * nix.

3. Розробка додатків GUI, що використовуються на стороні клієнта. Існують мови сценаріїв, спеціально розроблені для таких завдань, надають можливість PHP дуже зручно для WEB-майстрів, активно працюють з PHP і знають його досконало. Для цієї мети можна використовувати PHP-GTK.

Ви також можете створювати кросплатформні програми. PHP-GTK не входить в дистрибутив PHP, але є розширенням PHP.

PHP та Java – сучасні мови програмування, тому щороку виходять їх нові версії, які майже завжди сумісні з попередніми.

Враховуючи важливість зберігання та структурування даних користувача, в деяких випадках це є більш пріоритетним, ніж швидше, а також наявність загальної структури для всіх класів, що усуває необхідність у гнучкості бази даних, було вирішено використовувати базу даних MySQL, тому PHP має перевагу перед JAVA, тому з ORM легко працювати [24].

Як PHP так і Java підходять для реалізації поставленої задачі. Розглянемо переваги і недоліки кожної з мов (табл. 3.1).

Таблиця 3.1 – Порівняльна характеристика серверних мов програмування

Критерій	PHP	Java
1	2	3
Працює на своїй віртуальній машині	0	1
Призначений для WEB програмування	1	0

Продовження табл. 3.1

1	2	3
Легка інтеграція з JavaScript	1	0
Підтримка перевантаження опереторів	1	1
Сумарний коефіцієнт	3	2

Саме тому ця мова краще підходить для створення мобільного веб-додатка, оскільки має багато функцій, які значно полегшують процес розробки.

Дуже важливим є вибір інтегрованого середовища розробки. Серед найпоширеніших, які підтримують PHP і є безкоштовними: WebStorm, PhpStorm. Кожен з яких має свої особливості.

PhpStorm дійсно розуміє код. Підтримка базових фреймворків. PhpStorm ідеально підходить для роботи з Symfony, Drupal, WordPress, Zend

Framework, Laravel, Magento, Joomla!, CakePHP, Yii та інші фреймворки. Все, що потрібно для PHP.

PhpStorm глибоко аналізує структуру коду та розуміє ваш код, підтримуючи всі функції мови PHP як у нових, так і в застарілих проектах. Редактор підтримує автодоповнення та рефакторинг коду, запобігає помилкам «на льоту». Підтримка фронтенд-технологій.

У PhpStorm ви можете працювати з найновішими технологіями: HTML 5, CSS, Sass, Less, Stylus, CoffeeScript, TypeScript, Emmet і JavaScript [25].

будуть доступні рефакторинг, налагодження та модульне тестування. Завдяки функції редагування в реальному часі всі зміни можна відразу переглянути у браузері.

Вбудовані інструменти для розробників.

Одноманітні завдання зручно виконувати прямо в PhpStorm. IDE інтегрована з системами контролю версій, підтримує віддалене розгортання, бази даних і SQL, інструменти командного рядка, Docker, Composer, клієнт REST та багато інших інструментів.

PhpStorm = WebStorm + PHP + DB / SQL

PhpStorm включає в себе всю функціональність WebStorm, а також повну підтримку PHP, баз даних і SQL. Eclipse — це безкоштовне модульне інтегроване середовище розробки програмного забезпечення.

Eclipse — це в першу чергу повноцінна Java IDE, спрямована на групову розробку, має інструменти для роботи з системами контролю версій (підтримка CVS включена в Eclipse, активно розробляються кілька варіантів модулів SVN, є підтримка VSS та інші). Завдяки вільному характеру Eclipse є корпоративним стандартом для розробки програмного забезпечення Java у багатьох організаціях.

Друга мета Eclipse — служити платформою для нових розширень. Це C / C ++ Development Tools (CDT), розроблені інженерами QNX спільно з IBM, інструменти для підтримки інших мов від різних розробників. Багато розширень доповнюються менеджерами Eclipse для баз даних, серверів додатків та інших.

Однією зі зручних функцій phpStorm є те, що його можна використовувати

Результати аналізу середовищ розробки зведемо в табл. 3.2.

Таблиця 3.2 – Аналіз середовищ розробки

Критерій	Eclipse	PhpStorm
1	2	3
Автозаповнення з урахуванням контексту	1	1
Офіційна IDE для розробки на php	0	1
Підтримка php фреймворків	0	1

Продовження табл. 3.2

1	2	3
Підтримка unit-тестування	1	1
Сумарний коефіцієнт	2	4

3.2 Розробка API сервісу

API сервіс в даній системі виконує такі функції:

- аутентифікація користувача;
- авторизація користувача (надання прав);
- збереження та доступ до даних про анкету користувача;
- збереження та доступ до меню IVR;

Відповідно до функцій API сервіс забезпечує доступ до таких мутацій (Mutation) та запитів (Query):

Mutation loginUser – дозволяє отримати токен авторизації в системі, а також рефреш токен завдяки логіну та паролю.

Mutation tokenRefresh – дозволяє отримати новий токен авторизації завдяки рефреш токену, коли у поточного токену авторизації вийшов термін дії або став неактивний з інших причин.

Mutation registraterUser – дозволяє зареєструватися в системі, а також в результаті створення облікового запису надсилає електронне повідомлення з посиланням для підтвердження.

Рисунок 3.3 – Отримання листа на пошту

Mutation confirmRegister – дозволяє підтвердити реєстрацію по коду.

Mutation resendMail – дозволяє повторно відправити сповіщення у разі не вдалої спроби зареєструватися.

Query userProfile – дозволяє отримати дані користувача такі, як нікнейм та інформацію.

Mutation IVRCreateMutation – створює у БД запис про нове голосове меню IVR.

Mutation IVRUpdateMutation – оновлює у БД запис про голосове меню IVR.

Mutation IVRDeleteMutation – видаляє з БД запис про голосове меню IVR.

Доступ до цієї служби здійснюється за допомогою протоколу передачі даних Http.

Служба створена з використанням мови запитів GraphQL та фреймворку Laravel. В якості бази даних використовуються Mysql 8, Redis для кешованих даних і ElastickSearch для реєстрації.

Авторизація, реалізована за допомогою JSON Web Token (JWT), є відкритим стандартом RFC-7519 для створення маркерів доступу, які забезпечують безпечну передачу даних між двома сторонами.

Основним завданням JWT є передача даних у вигляді заяв між двома учасниками, важливим аспектом цього є стандартизація в простій формі, з необов'язковою перевіркою та шифруванням у простому форматі контейнера. Практичні застосування JWT включають:автентифікацію;

– авторизацію;
– сесії за допомогою клієнтської частини (“stateless” sessions);
– персональні дані клієнта.

Сеанси без збереження статусу є даними на стороні клієнта. Ключовим аспектом цього рішення є можливість використання цифрових підписів і шифрування для авторизації та захисту даних про стан сеансу, оскільки інформацію про клієнта неможливо підробити. Цифровий підпис використовується для захисту від підробки даних. Шифрування можна використовувати для захисту даних від використання третіми сторонами.

Сам токен складається з трьох елементів: заголовка, корисного навантаження та цифрового підпису. Перші два елементи мають певну структуру, а третій залежить від алгоритму, який використовується для шифрування. JWT може бути зашифрований у компактному представленні, відомому як JWS / JWE.

Компактна серіалізація — це захищене URL-адресою Base64 байтове кодування UTF-8 перших двох елементів JSON та інформації, необхідної для підписання або шифрування. Ці три елементи розділені крапками.

В результаті на сайті застосовуються токен в вигляді наведеному на рисунку 3.4.

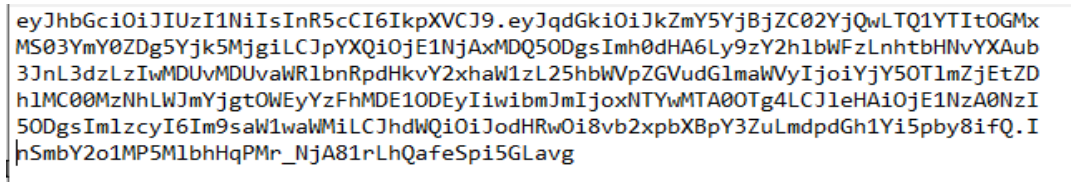

Рисунок 3.4 – Токен авторизації

В розкодованому стані хедер має вигляд наведений на рисунку 3.5.

Рисунок 3.5 – Хедер токену авторизації

Де параметр «alg» – це алгоритм, який використовується для генерації цифрового підпису.

Корисне навантаження має вигляд наведений на рисунку 3.6.

Рисунок 3.6 – Корисне навантаження токену авторизації

Тут кожен параметр має певне значення:

jti – ідентифікатор токена;

iss – визначає сторону, яка підписала токен;

aud – ресурс, який має право використовувати інформацію з токена;

exp – засвідчує момент часу, коли токен буде вважатись недійсним;

iat – засвідчує момент часу, коли токен був підписаний;

nbf – засвідчує момент часу, коли токен можна вважати дійсним;

nameidentifier – ідентифікатор користувача.

Для роботи з JWT токенами на серверній частині використано бібліотеку JWTLaravel.

Для отримання токена авторизації користувач повинен зробити http запит, який містить такі поля, як логін та пароль. Для верифікації токена використовується код який виконується кожен раз при надходженні запиту, так званий проміжний шар коду(«middleware»).

Як видно зі схеми конвеєр обробки запитів складається з невеликих блоків, в кожен з яких можна вставити власний код. Ковеєр middleware – це вхідна точка обробки кожного запиту, це набір компонентів та логіки, яка визначає чи потрібно передавати запит на обробку до наступного чи одразу відправити відповідь користувачу. Сюди також входить конвеєр маршрутизатора, який перевіряє чи відповідає Url запиту, якомусь із зареєстрованих маршрутів [26].

Розглянемо схему GraphQL API на рисунку 3.5.

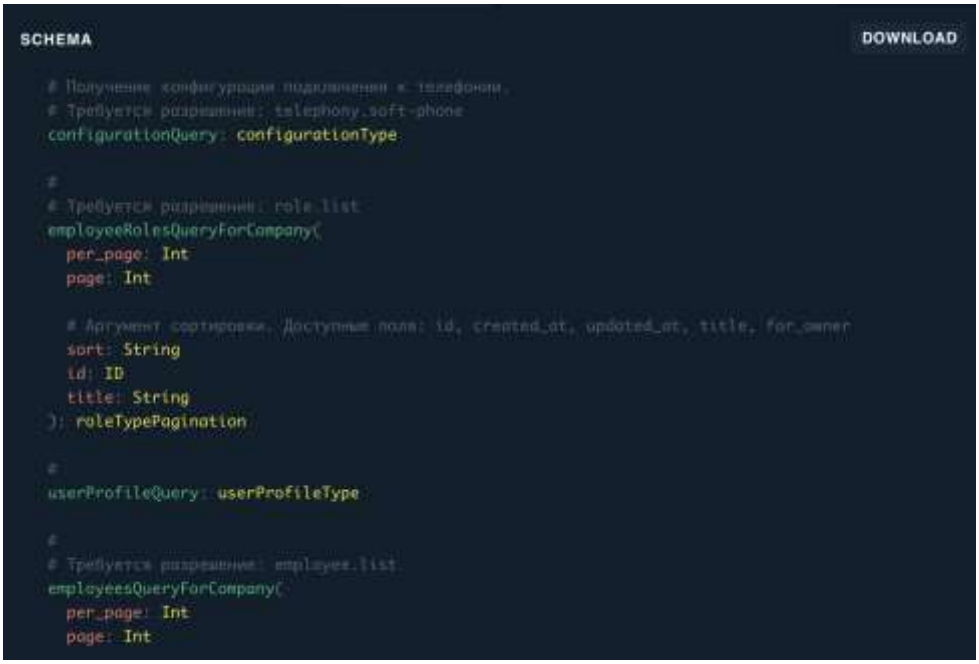

Рисунок 3.5 – Схема GraphQL запиту

Таким чином, розроблений сервіс виконує покладені функції, тобто доступ до даних, відповідає вимогам по безпеці.

3.3 Розробка клієнтської частини

Для реалізації клієнтської частини використано React.js. Створювати інтерактивні інтерфейси на React — приємно і просто. Достатньо описати, як частини інтерфейсу програми виглядають у різних станах. React буде своєчасно оновлювати їх, коли дані змінюються.

Створюйте інкапсульовані компоненти з власним станом, а потім об'єднуйте їх у складні інтерфейси користувача.

Оскільки логіка компонента написана на JavaScript, а не міститься в шаблонах, можна з легкістю передавати різні дані по всьому додатку і тримати стан поза DOM.

Нам не потрібно нічого знати про решту технологічного стека, тому є можливість розробляти нову функціональність на React, не змінюючи наявний код.

React також може працювати на сервері, використовуючи Node.js та на мобільних платформах, використовуючи React Native.

Управління даними в сучасних програмах є складним завданням. Більшість програм вимагає:

Окремі клієнти інтерфейсу для кількох платформ (веб, iOS тощо), кожна з яких має різні вимоги до даних

Бекенд, який обслуговує дані клієнтам з кількох джерел (Postgres, Redis тощо)

Складне керування станом і кеш-пам'яттю як для інтерфейсу, так і для бекенда.

Використовуючи GraphQL і Apollo, є можливість значно зменшити ці проблеми. Декларативна модель GraphQL допомагає створити послідовний, передбачуваний API, який можна використовувати для всіх своїх клієнтів. Під час додавання, видалення та міграції внутрішніх сховищ даних цей API не змінюється з точки зору клієнта.

Незважаючи на багато інших переваг, найбільшою перевагою GraphQL є досвід розробника, який він надає. Легко додавати нові типи та поля до вашого API, і так само легко для ваших клієнтів почати використовувати ці поля. Це допоможе вам швидко проектувати, розробляти та впроваджувати функції.

У поєднанні з платформою Apollo складні міркування, такі як кешування, нормалізація даних та оптимістична візуалізація інтерфейсу користувача, також стають простими.В даному випадку класи MatchHubConnection, RoomsHubConnection та ChatHubConnection слугують для встановлення з’єднання з відповідними сервісами, а також обробки повідомлень від цих сервісів.

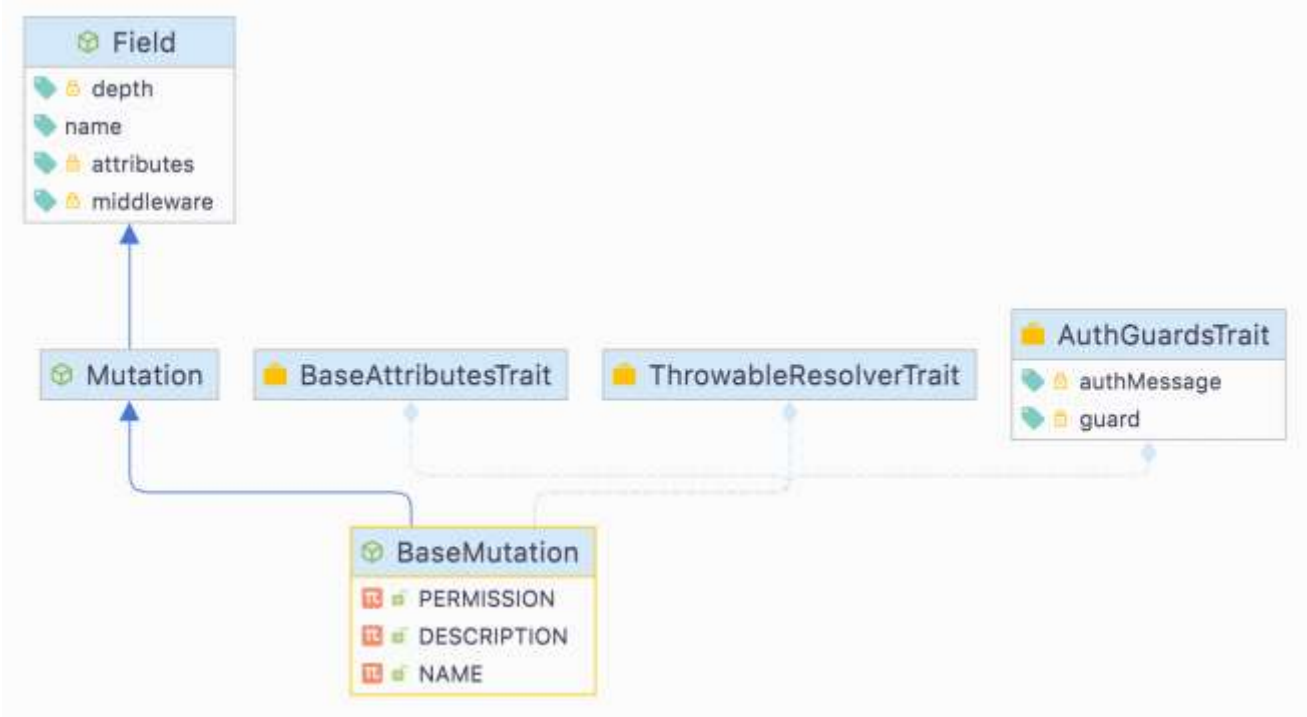

Рисунок 3.6 – Діаграма класів клієнта для роботи з сервером

Тобто в даному випадку реалізовано шаблон Model – view –controller(MVC).

Сутність даного шаблону проектування полягає у розбитті програми на три основні частини. Модель відповідає за дані і забезпечує інтерфейс доступу до них.

Контролер обробляє вхідні дані користувача та оновлює модель. В даному випадку контролером слугують класи для роботи з сервером, оскільки вся бізнес

логіка валідації та виконання ходу знаходиться на сервері, а також оновлення моделі відбувається в реальному часі.

Відображення – відповідає за представлення інформації на користувацькому інтерфейсі.

В даному випадку ключем є ідентифікатор користувача в форматі Json. Приклад формату даних, які зберігаються у сховищі Redis наведено на рисунку 3.9.

Рисунок 3.9 – Структура даних, які зберігаються у Redis

Таким чином, отриманий сервіс для покращення процесу створення голосового меню IVR в повному обсязі виконує поставлені задачі, так як у процесі розробки було обрано сучасні технології, які допомагають оптимізувати.

3.4 Висновки

У третьому розділі було проведено варіантний аналіз різноманітних технологій та засобів для розробки системи. У результаті якого було прийнято рішення: для серверної частини використати фреймворк laravel, у якості бази даних використати реляційну БД MySQL 8, у якості системи для кешування, збереження даних в реальному часі, та pub/sub функціональності використати

Redis та ElastickSearch для логування даних користувачів, зокрема помилок. Для клієнтської частини застосовано React з Apollo Client. Розробка проводиться за допомогою мови програмування php.

Визначено функціонал API сервісу та проведено аналіз процесу його розробки.

Визначено функціонал клієнту та проведено аналіз процесу його розробки.

Визначено функціонал створення голосового меню IVR та проведено аналіз процесу його розробки.

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Unit тестування

Модульне тестування, або юніт-тестування - процес в програмуванні, що дозволяє перевірити на коректність окремі модулі вихідного коду програми.

Ідея полягає в тому, щоб писати тести для кожної найбільш нетривіальної функції або методу. Це дозволяє досить швидко перевірити, чи не призвело чергову зміну коду до регресії, тобто до появи помилок у вже відтестованих місцях програми, а також полегшує виявлення та усунення таких помилок.

Коли говорять про якусь користь, більшість людей вважає що є якась метрика яка показує що без unit тестів було ось так, а з ними стало по іншому. Але таку метрику не так просто знайти чи внести на проект. З одного боку ми прагнемо підняти якість коду, а з іншого – unit тести покривають дуже малу і специфічну, ізольовану частину коду. Якщо говорити про ООП, то тести будуть перевіряти правильність роботи методів і класів, їхній зв’язок. Одними unit тестами ми не гарантуємо якість коду, але вони зменшать кількість дефектів спровокованих банальними помилками ізольованими в одному модулі. Це можна рахувати однією з метрик для вимірювання користі unit тестів.

Іншою метрикою можна назвати можливість легко змінювати систему. Якщо у вас є код повністю непокритий unit тестами – то вам страшно вносити будь-які зміни в систему, і тому набагато простіше скопіювати кусок коду і помістити поряд таку ж реалізацію. Якщо ж у вас є тести, то ви можете більш-менш безболісно змінити код і перевірити чи система продовжує працювати коректно. Представте що у вас є велика система, і як перевірити що вона працює правильно і так як задумано? Для цього її потрібно протестувати. Ви запускаєте весь цикл регресивного тестування залучивши до цього QA. Після внесення змін не можна бути впевненим, що система все так же якісно працює, тому треба знову все тестувати. Чим більше регресивного і ручного тестування – тим дорожчою являється розробка, а програміст пізніше отримує інформацію про зміни в системі.

Unit тести – це самий простий і близький до програміста спосіб сказати що система все ще якісна після того, як були внесені зміни [27].

Розробка системи без unit тестів буде швидша лише до певного етапу складності даної системи. Після досягнення цієї межі – кожне внесення зміни буде коштувати дорожче по часу, просто тому що ви не знаєте на скільки якісно працює система, у вас просто відсутній інструмент який може це сказати. І тому потрібно буде збільшувати кількість регресивно і ручного тестування. Побачити важливість тестів можна на тенденції створення нових мов і фреймворків, які зразу розробляються з вбудованими інструментами для unit тестів.

Якщо спочатку сісти й написати всю систему, а потім уже до неї писати тести – то так, час розробки збільшиться. При розробці системи ви уже її тестували: виводили в логи, проводили пошук багів. На цю роботу уже було потрачено багато часу. Якщо ж робити все по TDD, то спочатку ви пишете тест, визначаєте що ви хочете, а потім уже починаєте це реалізовувати. Виходить наступне:

1. фінальна реалізація буде набагато менша. У вас не включиться “режим архітектора” і не будете робити такі речі як реалізація на всі випадки життя. TDD каже, щоб ви робили тільки те що потрібно в цій задачі і нічого більше.
2. економиться час видумуванням способів як провалідувати код, не треба запускати всю системи доки не буде повністю завершена задача.
3. велику частину коду можна згенерувати з тесту. Сучасні IDE дуже розвинені. Не потрібно про це забувати. Наприклад в Java або .NET можна згенерувати частину коду по певному сценарію. Це економить багато часу.

Unit тести зменшують кількість дефектів від 40% до 90%, хоч при цьому і збільшується початковий час розробки. Потрібно навчитись писати тести, писати код так щоб його можна було тестувати. На початковому етапі часу буде тратитись на 15-30% більше, але і через пів року можна буде вносити зміни так же легко. Це довготермінові інвестиції.

Вони прості й зрозумілі. Якщо ви через місяць дивитесь на свій тест або тест свого колеги і думаєте що краще я його видалю і напишу новий – це уже неправильно. Якщо дивитесь на код і розумієте що тест ще складніший – це теж не те що треба.

Характеристика Unit тестів. Вони повинні забезпечувати Quality gate – захищати вас від допущених помилок, якщо ви пішли й змінили щось в чому не дуже розібралися. Для цього вони повинні бути ізольованими.

Найменування тестів. Вони повинні казати що "впали" та чому. Тести потрібно ділити на маленькі ізольовані частини.

Unit тест який не можна відрізнити хто з команди написав. Повинна бути консистентність. Уникати ситуацій коли назва тесту написана не інформативно і не описує його роботу. Для цього потрібно залучити Code review і статичний аналіз коду. Тести потрібно перевіряти так само як і основний код.

Unit тести повинні бути без side ефектів. Вони повинні бути незалежні і запускатися окремо.

Читабельність. Коли є вхідні параметри, виконусь ось це і результат такий. Це створює автодокументацію коду і допомагає робити code reveiw більше ефективним. Review юніт тестів разом з кодом допомагає зрозуміти які use кейси були покриті, чи все реалізовано.

Чисті тести. Все з основ чистого коду. Повинен читатися легко, навіть якщо тестує складний код.

Тести повинні бути швидкими. Вони не повинні підключатися до бази даних, щось отримувати з мережі чи читати з файлової системи. Навіть на дуже великому проекті всі тести повинні проходити не довше декількох хвилин.

Тести повинні перевіряти ЩО робить код а не ЯК він це робить. Щоб була можливість код змінити або порефакторити.

Unit тести повинні бути іменно Unit тестами а не інтеграційними, наприклад. Тулзи дозволять писати інтеграційні тести, які ми називаємо Unit тестами, а потім жаліємось чому вони такі нестабільні.

Намагатися зберігати пірамуду тестування. Unit тестів повинно бути багато, а тестів вищого рівня менше. Часто на проектах пишуть багато інтеграційних тестів і мало unit – це антипатерн, перевернута піраміда.

Дана метрика лише допомагає і ніколи не повинна визначати. Це не повинна бути метрика менеджера чи користувача. Дана метрика повинна допомагали лише розробнику зрозуміти працює він добре чи погано. Вона ніколи не визначає чи тести написані добре, навіть якщо покриття 100%. Code coverage – це інструмент розробника для розробки, а не менеджера для оцінки чого небудь.

По хорошому, закінчивши задачу, розробник повинен прогнати code coverage і той повинен показати йому сценарії які він не врахував.

Цифра покриття, по суті, нічого не значить. Значення має звіт, який ми читаємо і розуміємо. Він показує куски коду які не покриті. Можна побачити що якийсь кусок коду критичний і потребую тесту, інший ж не критичний і не потребує. Покриття повинне показувати код який ще не покрили. Відсоток покриття може становити і 90%, але є 10% критичного коду який не покритий. В цьому випадку використання code coverage неправильне.

Юніт тестування не являється чимось особливим, чимось стороннім, додатковим. Під час розробки ми постійно тестуємо код, тому що так побудований спосіб мислення і процес розробки. При цьому ми постійно придумуємо test cases, аналізуємо як код виконує одну чи іншу задачу і постійно їх виконуємо порівнюючи результат з тим що ми повинні отримати. Практично, різниця між автоматичним юніт тестом і тим тестуванням яке ми робимо під час розробки в своїй голові полягає лише в тому що ми його кудись записуємо – unit test як код. Немає ніякого іншого додаткового процесу написання unit test, він завжди являється невід’ємною частиною розробки.

PHPUnit – це система для юніт-тестування програм, написаних мовою PHP [28]. Під "юніт" розуміються невеликі блоки коду, наприклад, окремі методи класу. Тобто. можна протестувати спосіб працездатність в автоматичному режимі. Коли програма досить велика, що містить багато класів, методів і тим більше якщо планується подальше його розширення, варто зайнятися тестуванням і в цьому

допоможе PHPUnit. Він є окремою бібліотекою класів, які потрібно підключити при створенні свого тесту.

Для вирішення нашого завдання було вирішено написати декілька unit-test для перевірки коректності роботи критично важливих етапів алгоритму.

Було розроблено тести для наступних модулів:

AuthModule – модуль для авторизації, аутентифікації користувача.

IVRModule – модуль для створення, редагування та видалення налаштувань голосового меню IVR та його сценарію.

NotificationModule – модуль відправки сповіщень, шляхом відправки email повідомлень про важливі події, що стосуються стану користувача, наприклад успішна чи невдала реєстрація, оновлення IVr тощо.

Було розроблено наступні тести:

Тест 1 – перевіряє чи правильно відпрацьовує модулі AuthModule та NotificationModule при роботі зі зміною пароля.

Під час тестування створився mock-об’єкт користувача. Перестворилась база даних (для чистоти теста), перестворився сам додаток, встановились необхідні Permission для користувача. Відбувся логін у систему та виконався запит до мутації, що дозволяє змінити користувацький пароль. Після цього відбулася перевірка на успішне проходження основних етапів, а саме:

1. Перевірка на корректну аутентифікацію
2. Перевірка на те, що користувач отримав необхідні доступи
3. Перевірка на те, що користувач має право виконувати мутацію
4. Перевірка на те, що пароль було змінено успішно
5. Перевірка на те, що користувачеві було відправлено листа

Приклад лістингу програмного коду наведено нижче на рисунку 4.1.

```php
public function test_employee_get_notification_with_his_new_password_on_updating_user(): void
{
    $manager = $this->loginAsEmployeesManager();
    $updatingEmployee = User::factory()->create();
    CompanyUser::query()->create(
        [
            'user_id' => $updatingEmployee->id,
            'company_id' => $manager->company->id
        ]
    );

    Notification::fake();

    $this->query($updatingEmployee->id);
    $updatingEmployee->refresh();

    Notification::assertNotSentTo(
        new AnonymousNotifiable(),
        notification: SendPasswordNotification::class,
        function ($notification, $channels, $notifiable) use ($updatingEmployee) {
            return $notifiable->routes['mail'] === $updatingEmployee->getEmailString();
        }
    );

    $this->data['send_email'] = 'true';
    $this->data['password'] = 'new-password';
    $this->data['email'] = 'new-email@gmail.com';

    Notification::fake();

    $this->query($updatingEmployee->id);
```

Рисунок 4.1 – Лістинг коду теста 1

Тест 2 – даний тест перевіряє чи отримаємо ми помилку при спробі видалити користувача при відсутності на такую дію прав. Під час тестування створився mock-об’єкт користувача (якого хочемо видалити) та ще одного користувача (який бажає видалити). Перестворилась база даних (для чистоти теста), перестворився сам додаток, не встановились необхідні Permission для користувача, що бажає здійснити видалення. Відбувся логін у систему та виконався запит до мутації, що дозволяє здійснити видалення іншого користувача лише за нявності коректного доступу. Після цього відбулася перевірка на успішне проходження основних етапів, а саме:

1. Перевірка на корректну аутентифікацію
2. Перевірка на те, що користувач не отримав необхідні доступи
3. Перевірка на те, що користувач не має права виконувати мутацію
4. Перевірка на те, що жоден користувач не був видалений

Код лістингу теста наведено на рисунку 4.2 нижче.

```
public function test_cant_delete_employee_by_not_auth_user(Company $company = null): void
{
    $deletedEmployee = User::factory()
        ->withCompany($company)
        ->create();

    $result = $this->query($deletedEmployee->id)
        ->assertOk();

    $this->assertGraphQlUnauthorized($result);
}

protected function query(int $id): TestResponse
{
    $query = sprintf(
        format: 'mutation { %s (ids: [%s]) }',
        ...values: self::MUTATION,
        $id,
    );

    return $this->postGraphQL(['query' => $query]);
}
```

Рисунок 4.2 – Лістинг коду теста 2

Тест 3 – перевіряє чи не має можливості створити нове голосове меню неавторизований та не аутентифікований користувач. Під час тестування не створився mock-об’єкт користувача. Перестворилась база даних (для чистоти теста), перестворився сам додаток. Не відбувся логін у систему та виконався запит до мутації, що дозволяє здійснити створення голосового меню IVR. Після цього відбулася перевірка на успішне проходження основних етапів, а саме:

1. Перевірка на те, що користувач не має доступу до даної мутації

2. Перевірка на те, що користувач не отримав необхідні доступи
3. Перевірка на те, що користувач не має права виконувати мутацію
4. Перевірка на те, що жоден IVR не був створений

Код лістингу цього теста наведено на рисунку 4.3.

```
public const MUTATION = IVRCreateMutation::NAME;

protected array $data = [
    'name' => 'Test ivr name',
    'description' => 'Test ivr description',
];

public function test_cant_create_ivr_by_guest(): void
{
    $result = $this->query()
        ->assertOk();

    $this->assertGraphQlUnauthorized($result);
}
```

Рисунок 4.3- Лістинг коду теста 3

Тест 4 - перевіряє чи не має можливості створити нове голосове меню неавторизований, але аутентифікований користувач. Під час тестування створився mock-об’єкт користувача, що бажає здійснити створення нового меню IVR. Перестворилась база даних (для чистоти теста), перестворився сам додаток. Відбувся логін у систему та виконався запит до мутації, що дозволяє здійснити створення голосового меню IVR. Після цього відбулася перевірка на успішне проходження основних етапів, а саме:

1. Перевірка на те, що користувач має доступ до даної мутації
2. Перевірка на те, що користувач не отримав необхідні доступи

3. Перевірка на те, що користувач не має права виконувати мутацію
4. Перевірка на те, що жоден IVR не був створений

Код лістингу цього теста наведено на рисунку 4.4.

```
public function test_cant_create_ivr_by_user_without_permission(): void
{
    $this->loginAsUser();

    $result = $this->query()
        ->assertOk();

    $this->assertGraphQlUnauthorized($result);
}
```

Рисунок 4.4 – Лістинг коду теста 4

Тест 5 - перевіряє чи має можливость створити нове голосове меню авторизований та аутентифікований користувач. Під час тестування створився mock-об’єкт користувача, що бажає здійснити створення нового меню IVR. Перестворилась база даних (для чистоти теста), перестворився сам додаток. Відбувся логін у систему та виконався запит до мутації, що дозволяє здійснити створення голосового меню IVR. Після цього відбулася перевірка на успішне проходження основних етапів, а саме:

1. Перевірка на те, що користувач має доступ до даної мутації
2. Перевірка на те, що користувач отримав необхідні доступи
3. Перевірка на те, що користувач має право виконувати мутацію
4. Перевірка на те, що було створено новий IVR

Код лістингу цього теста наведено на рисунку 4.5.

```php
public function test_can_create_ivr_by_user_with_permission(): void
{
    $company = Company::factory()->create();
    $user = User::factory()->withCompany($company)->create();
    Domain::factory()->withCompany($company)->create();

    $this->assertDatabaseMissing( table: IVR::TABLE, $this->data);

    $this->loginAsUser($user)
        ->assignRole(
            $this->generateRole(
                name: 'Locale IVR user',
                [
                    IVRCreatePermission::KEY,
                ]
            )
        );

    $result = $this->query()
        ->assertOk();

    $response = $result->json( key: 'data.' . static::MUTATION);

    $this->assertEquals($this->data, $response);
    $this->assertDatabaseHas( table: IVR::TABLE, $this->data);
}
```

Рисунок 4.5 – Лістинг коду теста 5

Для виконання наступних тестів необхідно провести підготовчу роботу, а саме: реалізувати декілька методів для комплексної перевірки перевірки стану IVR меню та залежних модулів.

Для початку написати методи перевірки, що у всіх базах даних відсутня будь-яка інформація про IVR.

Лістинг коду наведено на рисунку 4.6.

```php
public function assertCommonSettingsNotAttached(): void
{
    $this->assertDatabaseMissing( table: IVR::TABLE, [
        'id' => $this->data['id'],
        'name' => $this->data['name'],
        'description' => $this->data['description'],
        'ivr_number' => $this->data['ivr_number'],
        'domain_id' => $this->data['domain_id'],
        'attempts' => $this->data['attempts'],
        'timeout' => $this->data['timeout'],
        'return_code' => $this->data['return_code'],
        'record' => $this->data['record'],
        'language_change' => $this->data['language_change'],
    ]);
}
```

Рисунок 4.6 – Лістингу коду, який перевіряє відсутність IVR у БД

Далі необхідно реалізувати зворотній метод – який перевірить коректне зберігання введених користувачем даних.

Лістинг коду наведено на рисунку 4.7.

```php
public function assertCommonSettingsAttached(): void
{
    $this->assertDatabaseHas( table: IVR::TABLE, [
        'id' => $this->data['id'],
        'name' => $this->data['name'],
        'description' => $this->data['description'],
        'ivr_number' => $this->data['ivr_number'],
        'domain_id' => $this->data['domain_id'],
        'attempts' => $this->data['attempts'],
        'timeout' => $this->data['timeout'],
        'return_code' => $this->data['return_code'],
        'record' => $this->data['record'] ? 1 : 0,
        'language_change' => $this->data['language_change'] ? 1 : 0,
    ]);
}
```

Рисунок 4.7 - Лістингу коду, який перевіряє наявність IVR у БД

Також необхідна можливість у зручній формі перевірити чи правильно зберіглися переводи аудио та інших сутностей, що стосуються IVR. Реалізуємо функцію complexAssertLocalesAttached, відповідальність якої буде полягати у такій перевірці.

Лістинг коду наведено на рисунку 4.8.

```
public function complexAssertLocalesAttached(): void
{
    if ($this->data['language_change']) {
        $this->assertDatabaseCount(
            table: IVRLanguage::TABLE,
            count($this->data['languages']),
            connection: DbConnections::ASTERISK_CFG
        );
    } else {
        $this->assertDatabaseCount(
            table: IVRLanguage::TABLE,
            count: 0,
            connection: DbConnections::ASTERISK_CFG
        );
    }

    foreach ($this->data['languages'] as $language) {
        $this->assertDatabaseHas( table: Locale::TABLE, [
            'slug' => $language['locale']
        ]);

        $this->assertDatabaseHas( table: IVRLocale::TABLE, [
            'locale' => $language['locale'],
            'ivr_id' => $this->IVR->id,
            'main' => $language['is_main'],
            'key_number' => $language['key_number'],
        ]);

        if ($this->data['language_change']) {
            $this->assertDatabaseHas(
                table: IVRLanguage::TABLE,
```

Рисунок 4.8 – Лістинг коду функції complexAssertLocalesAttached

Тест 6 - перевіряє чи має можливость оновити уже існуюче голосове меню авторизований та аутентифікований користувач. Під час тестування створився mock-об’єкт користувача, що бажає здійснити оновлення меню IVR, також було створено mock-об’єкт самого IVR меню. Перестворилась база даних (для чистоти теста), перестворився сам додаток. Відбувся логін у систему та виконався запит до мутації, що дозволяє оновити голосовоу меню IVR. Після цього відбулася перевірка на успішне проходження основних етапів, а саме:

1. Перевірка на те, що користувач має доступ до даної мутації
2. Перевірка на те, що користувач отримав необхідні доступи
3. Перевірка на те, що користувач має право виконувати мутацію
4. Перевірка на те, що перед виконанням тесту у БД існує лише 1 голосове меню IVR.
5. Перевірка на те, що перед виконанням тесту у БД існує залежна інформація про аудиофайли тестового IVR
6. Перевірка на те, що було корректно оновлено існуючий IVR з усіма залежностями

Завдяки такому тесту у нас є можливість перевірки цілісного стану БД за рахунок опису користувацьких даних у масиві $data. Та використовуючи власне написані функції тестування комплексного стану додатку. Як перевірка стану додатку до виконання тесту, так і перевірка стану додатку після виконання тесту. Ці методі враховують лише користувацькі дані та дають впевненість у тому, що мутація виконала лише необхідні дії по наповненню таблиць бази даних за рахунок запам’ятовування стану нашого додатку «до» та «після».

Код лістингу цього теста наведено на рисунку 4.9.

```php
public function test_can_update_ivr_by_user_with_permission(): void
{
    $this->firstUser->assignRole($this->generateRole( name: 'Locale IVR user', [IVRUpdatePermission::KEY,]));

    $this->loginAsUser($this->firstUser);

    $this->complexAssertAnnouncementNotAttached();
    $this->assertCommonSettingsNotAttached();
    $this->complexAssertLocalesNotAttached();
    $this->complexAssertFailoversNotAttached();

    $this->updateAndCheck();

    $this->data['language_change'] = true;
    $this->data['languages'][0]['key_number'] = '2';
    $this->data['languages'][] = [
        'is_main' => false,
        'locale' => 'en',
        'key_number' => '1'
    ];

    $this->updateAndCheck( needDump: true);

    $this->data['languages'][0]['key_number'] = null;
    unset($this->data['languages'][1]);
    $this->data['language_change'] = false;

    $this->updateAndCheck();

    $this->data['name'] = 'updated name';
    $this->data['description'] = 'updated description';
```

Рисунок 4.9 – Лістинг коду теста 6

Результат виконання модульних тестів наведено на рисунку 4.10.

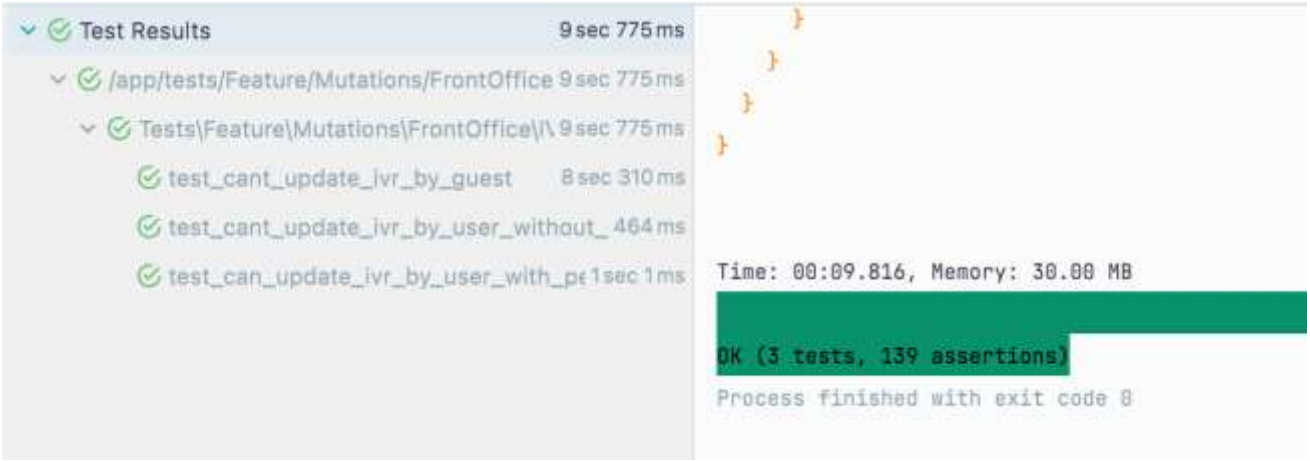

Рисунок 4.10 – Результат виконання модульних тестів

4.2 Розробка інструкції користувача

Щоб створити власний IVR, необхідно зареєструватись у системі (рис. 4.11), для цього потрібно виконати такі дії:

1. Ввести свої прізвище, ім’я, по-батькові, контактний номер телефону та email у відповідні поля вводу.
2. Придумати надійний пароль, ввести його в одноіменне поле.
3. Повторити пароль у наступному полі.
4. Погодитись з політикою обробки персональних даних (поставити галочку)
5. Натиснути кнопку «Зареєструватись і продовжити»

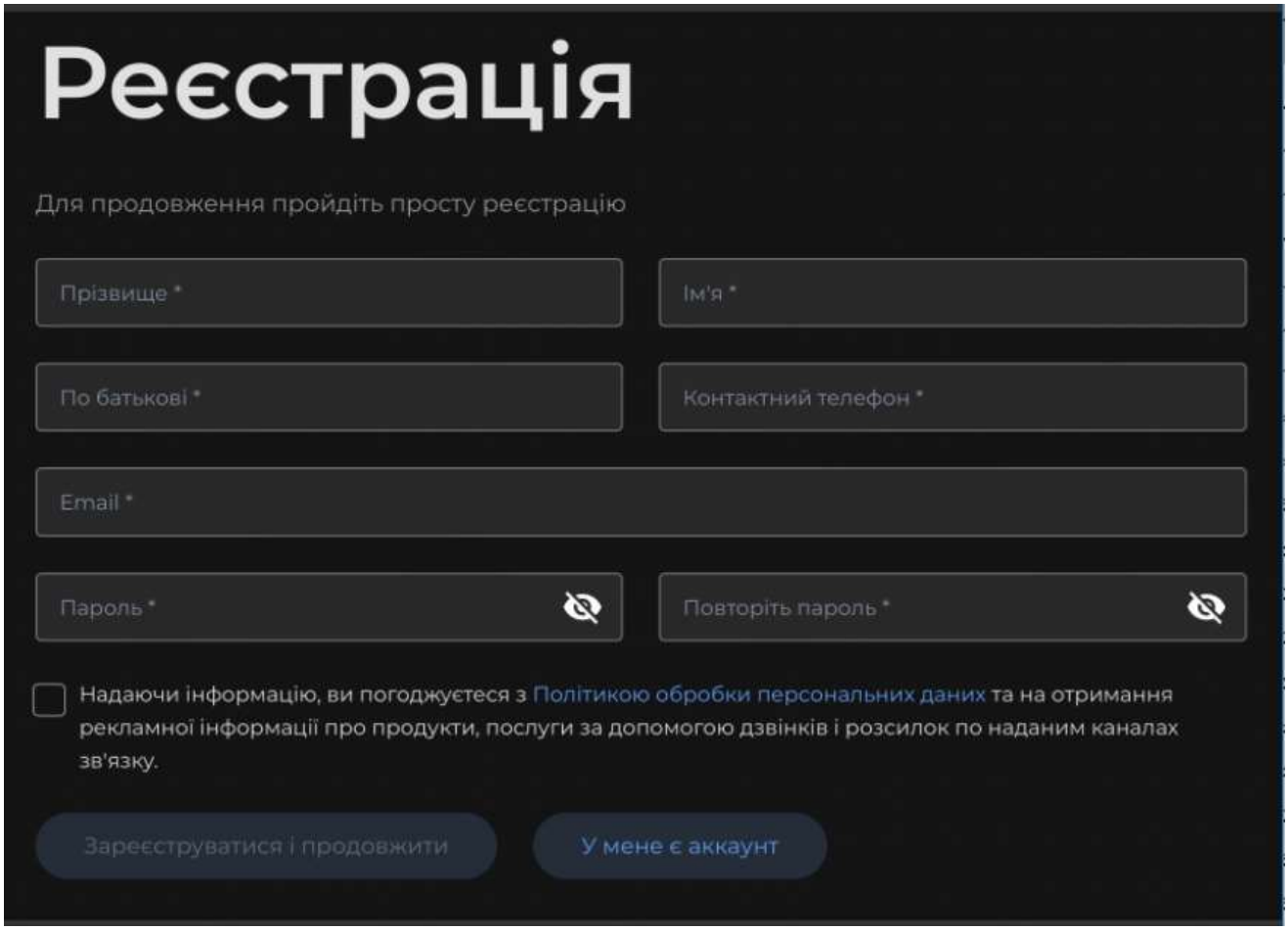

Рисунок 4.11 – Реєстрація

Щоб увійти в систему (рис. 4.12 – 4.13), потрібно виконати такі дії:

1. Заповнити поле «Ваш email» і «Пароль»
2. Натиснути кнопку «Увійти».
3. Дочекатись повідомлення про успішну авторизацію.

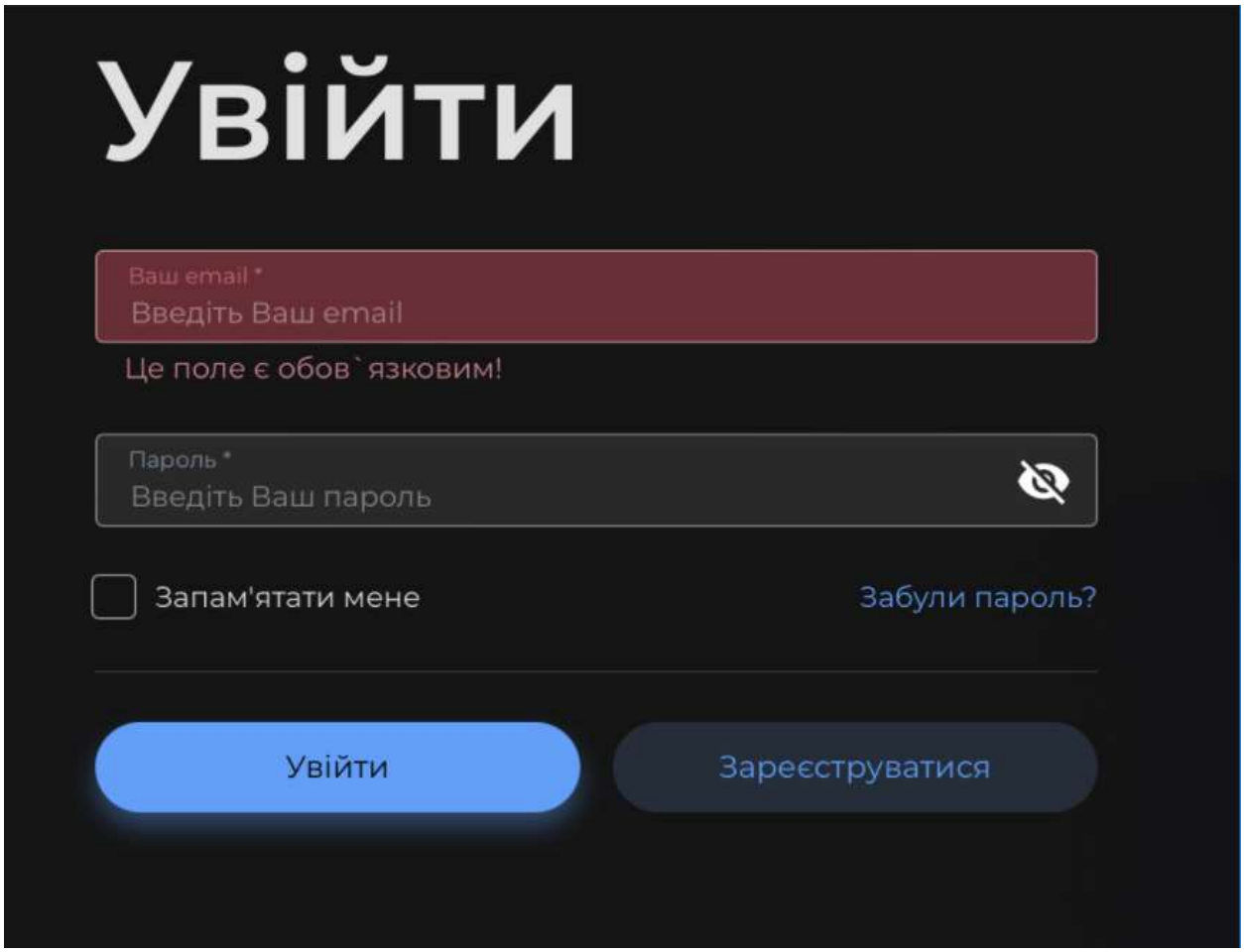

Рисунок 4.12 – Вхід у систему

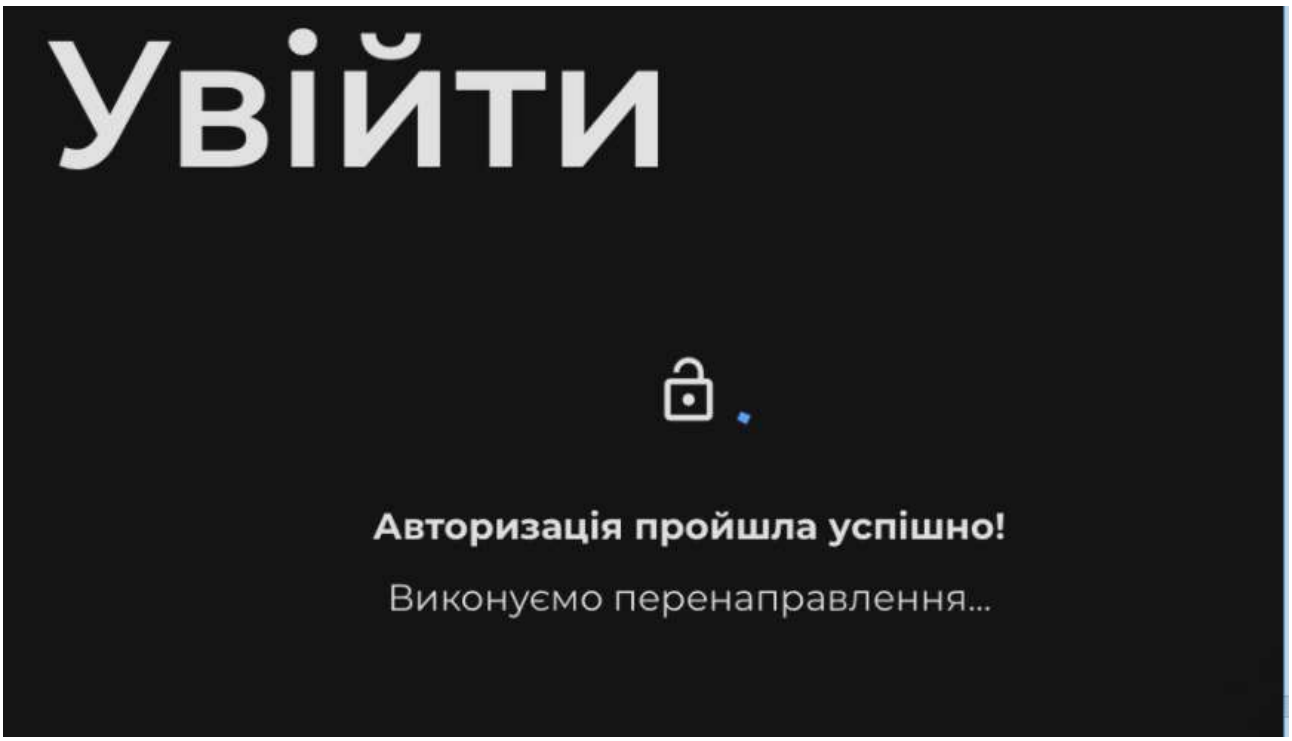

Рисунок 4.13 – Повідомлення про успішну авторизацію

Після входу в систему, з’являється вікно з вибором шаблону (рис. 4.14). Можна переглянути детальну інформацію про нього чи зайти в налаштування.

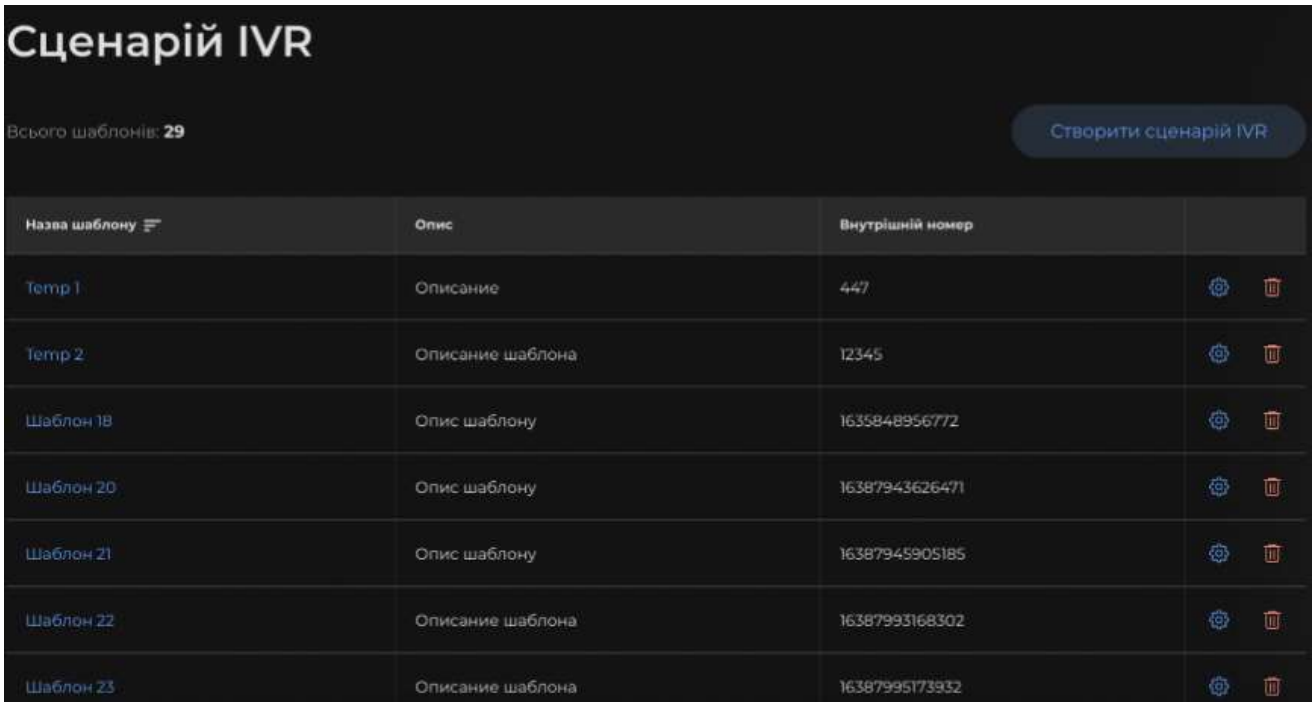

Рисунок 4.14 – Вибір шаблону

Для того, щоб створити власний шаблон сценарію IVR (рис. 4.15), потрібно виконати такі дії:

1. Натиснути на кнопку «Створити сценарій IVR».
2. Ввести назву шаблону.
3. Обрати домен.
4. За бажанням, додати опис свого шаблону.
5. Натиснути кнопку «Додати/редагувати мову».
6. Обрати бажану мову голосового помічника.
7. Натиснути кнопку «Далі».

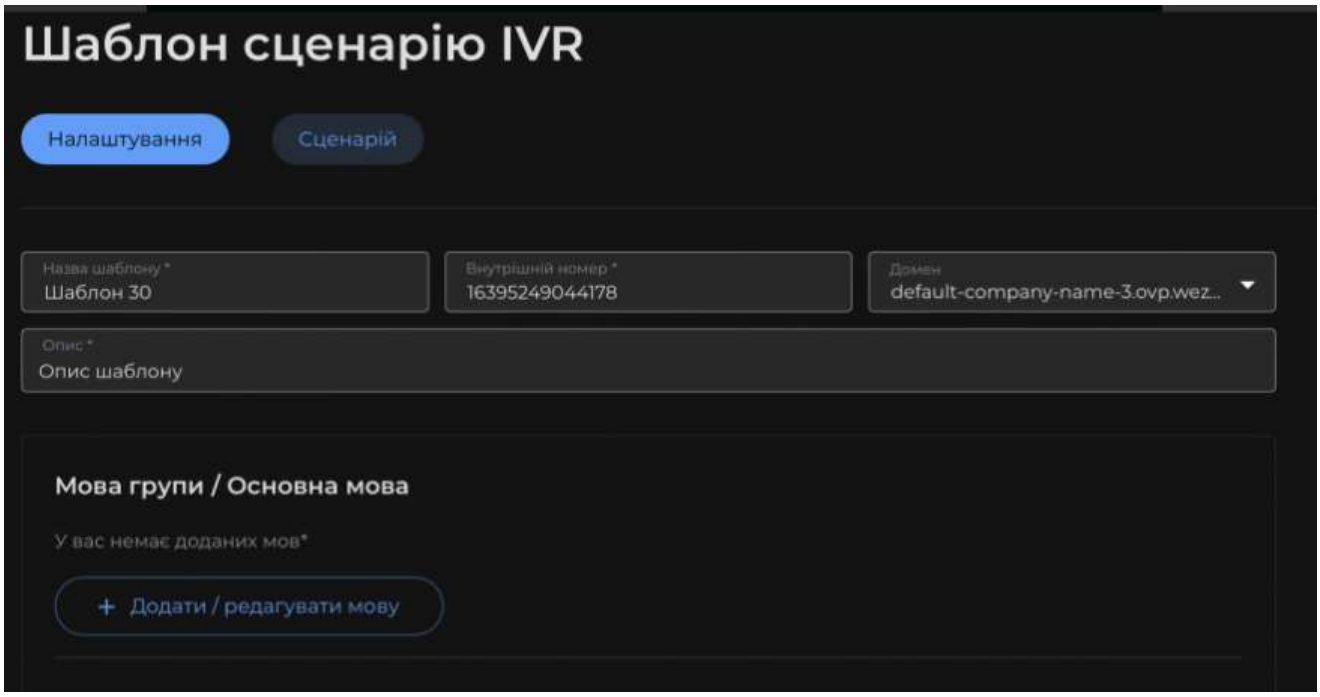

Рисунок 4.15 – Налаштування шаблону сценарію

Якщо проігнорувати вказівки та не обрати мову – не буде можливості для користувача створити сценарій (рис. 4.16)

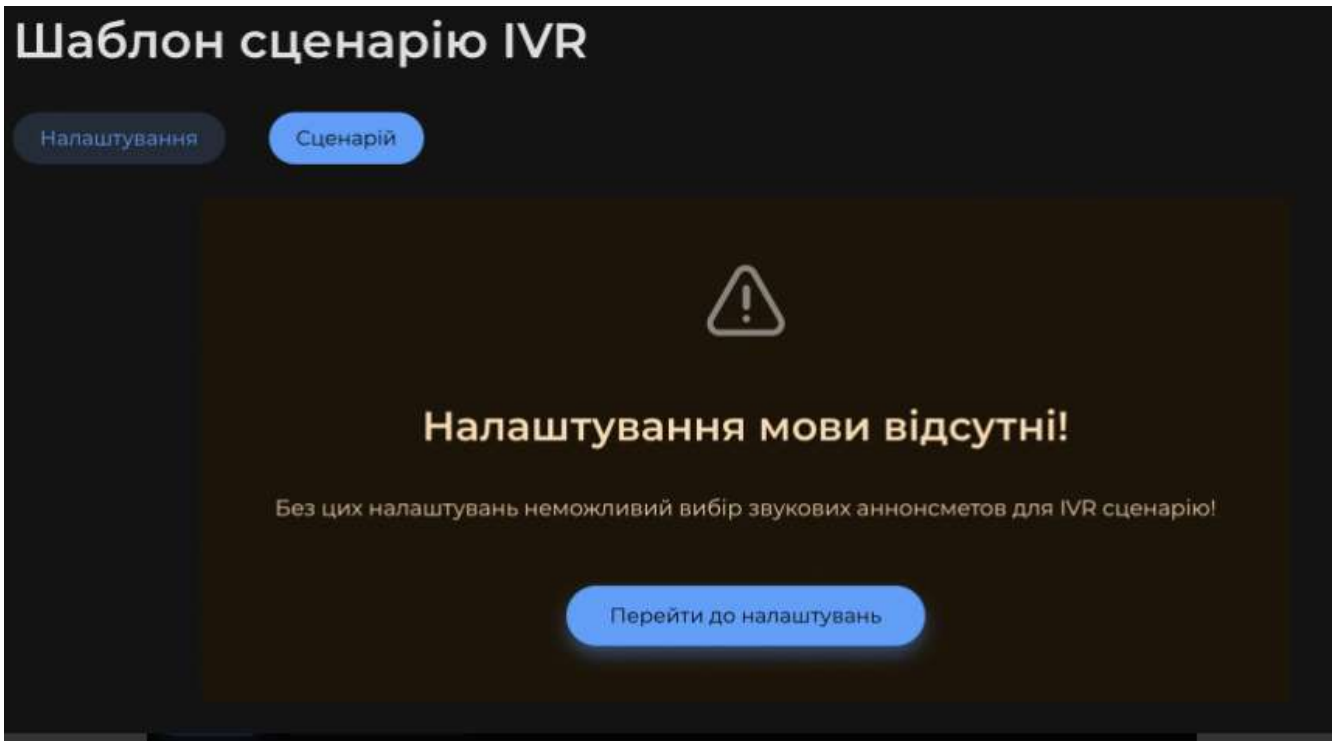

Рисунок 4.16 – Попередження при необраній мові

Важливим кроком у створенні IVR є налаштування утримання (рис. 4.17), тому в наступному вікні необхідно виконати такі дії:

1. Обрати час очікування в секундах.
2. Обрати або додати аудіо для перевищення очікування.
3. Обрати рішення при таймауті та перевищенні кількості спроб введення.
4. Натиснути кнопку «Далі».

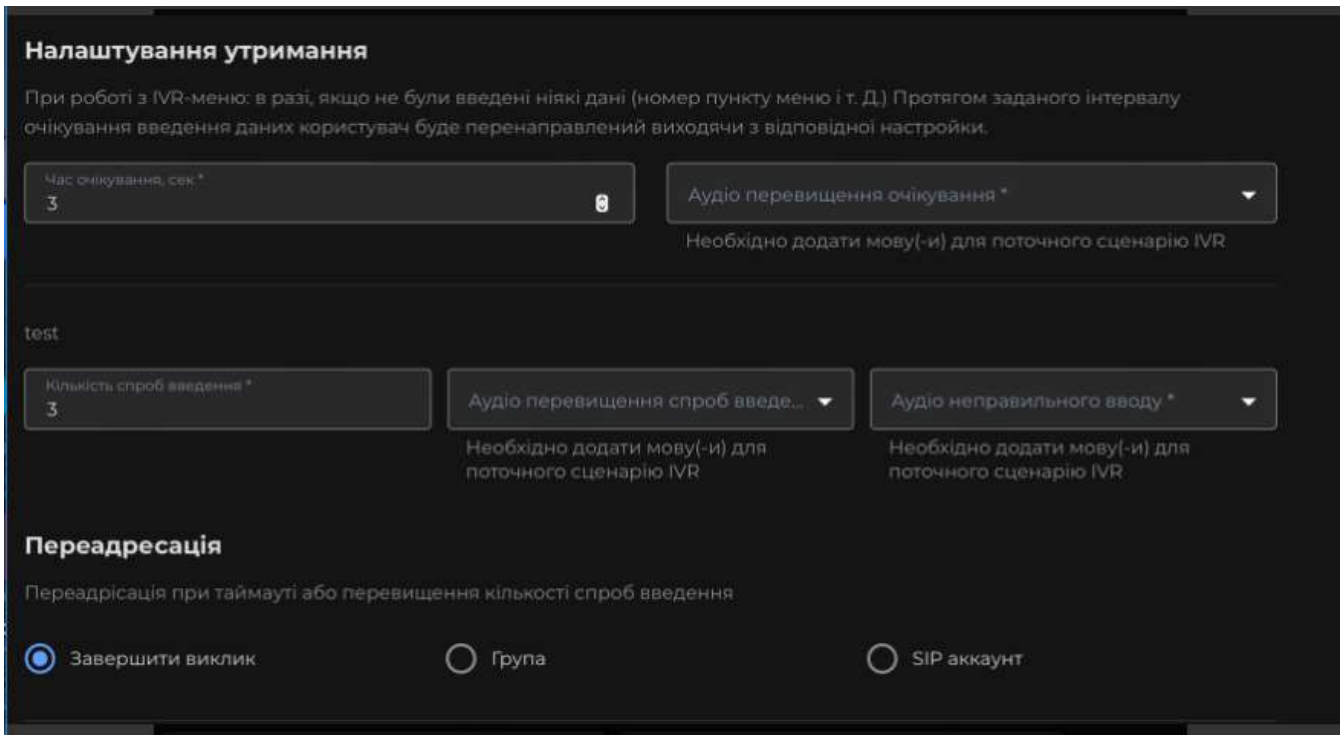

Рисунок 4.17 – Налаштування утримання

Наступним з’явиться вікно «Повернення в меню» (рис. 4.18), потрібно виконати такі дії:

1. Обрати зі списку аудіо при додатковому наборі номеру.
2. Обрати зі списку аудіо неправильного вводу.
3. Натиснути кнопку «Зберегти».

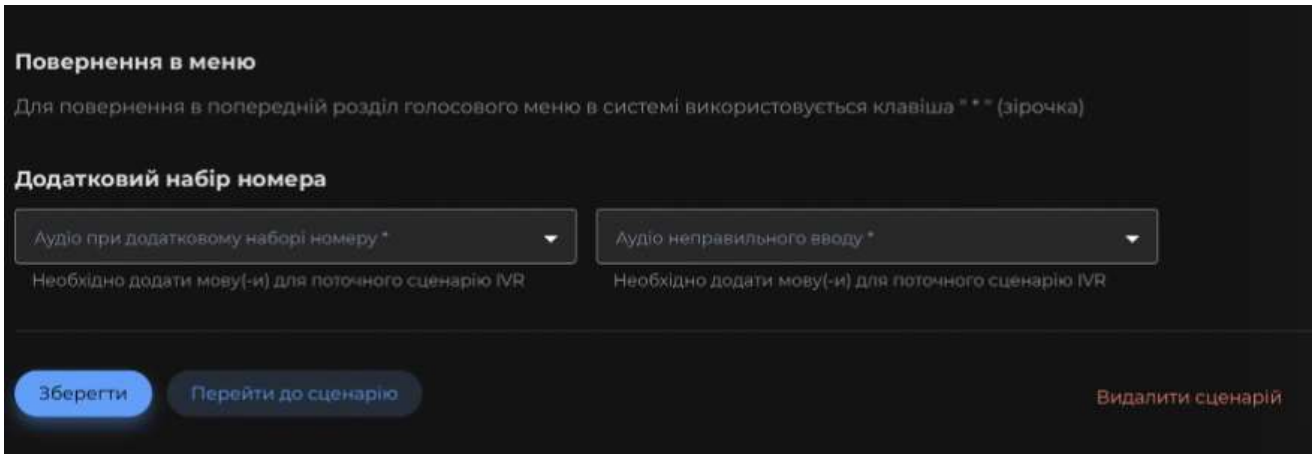

Рисунок 4.18 – завершення налаштування шаблону

Якщо пропустити вибір аудіо, система не збереже налаштування, вкаже на не заповнені обов’язкові поля(рис. 4.19)

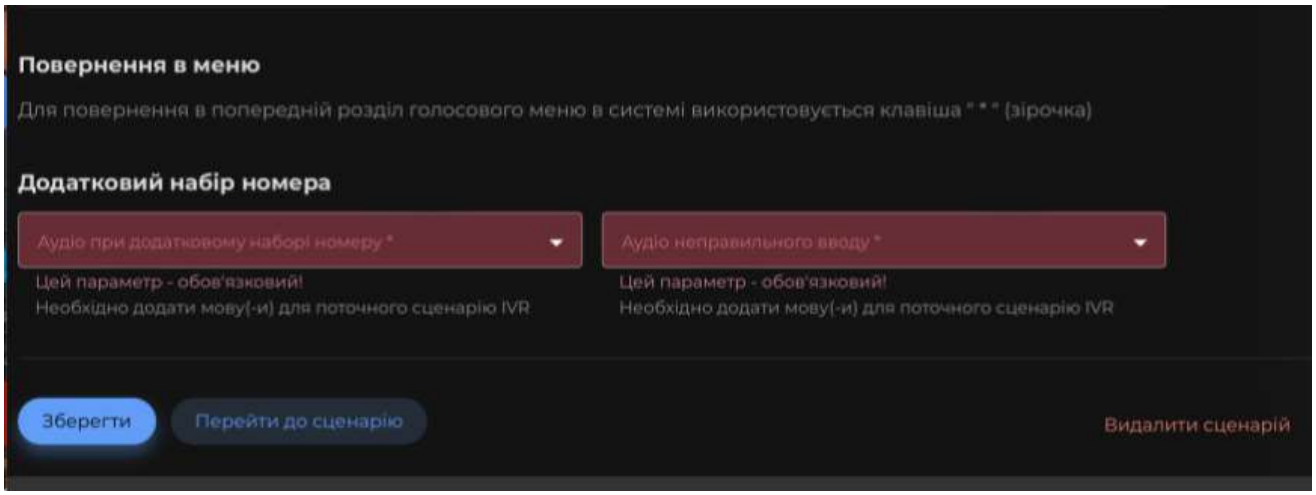

Рисунок 4.19 – Валідація введених даних

Для створення власного сценарію IVR, слід дотримуватись наступних кроків:

1. У вікні «Створення сценарію IVR» натиснути кнопку «Сценарій»;
2. Обрати звукове привітання, натиснувши на відповідну кнопку;
3. Підтвердити вибір звукового привітання, натиснувши на кнопку «Підтвердити»;
4. Натиснути кнопку «+ Пункт голосового меню» (рис.20);

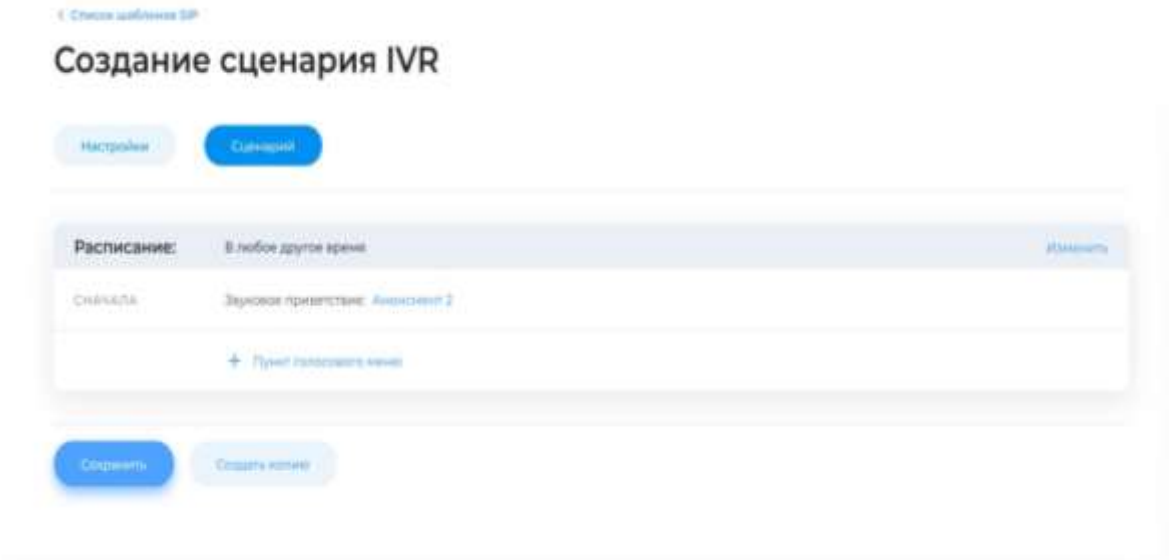

Рисунок 4.20 – Створення сценарію IVR

1. Натиснути кнопку «+ Пункт голосового меню» (рис.4.21);

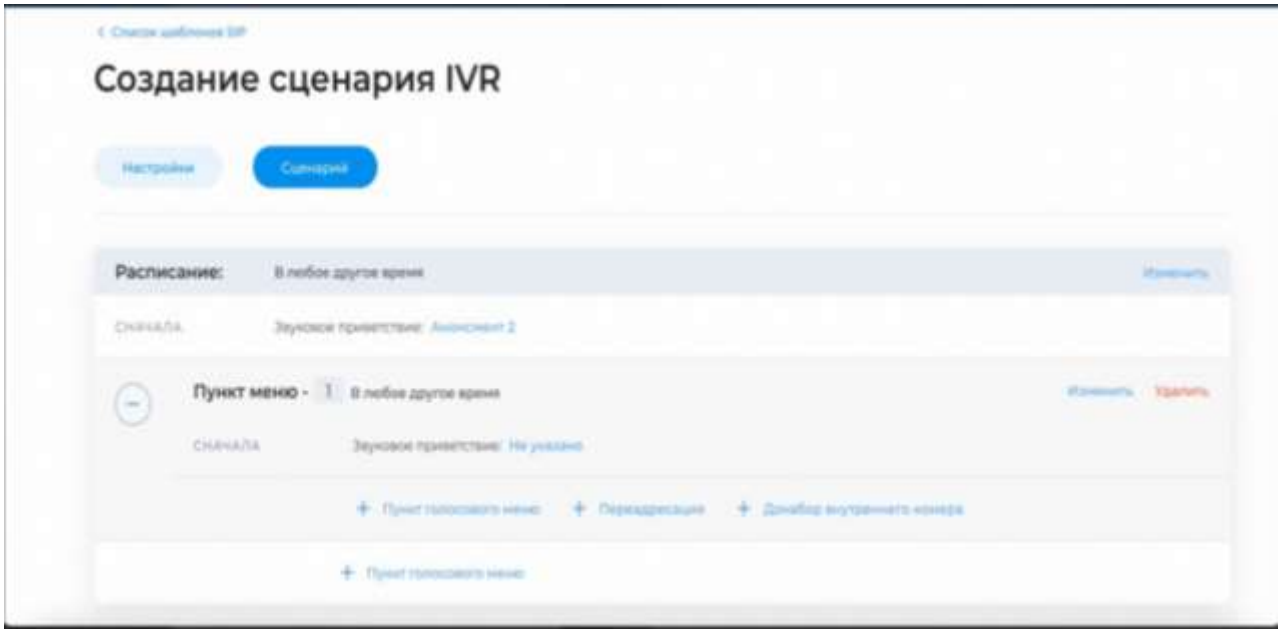

Рисунок 4.21 – Додавання пункту голосового меню

2. Аналогічно до П.2 обрати для підпункту голосове привітання (рис.4.22);
3. Підтвердити вибір кнопкою «Підтвердити»

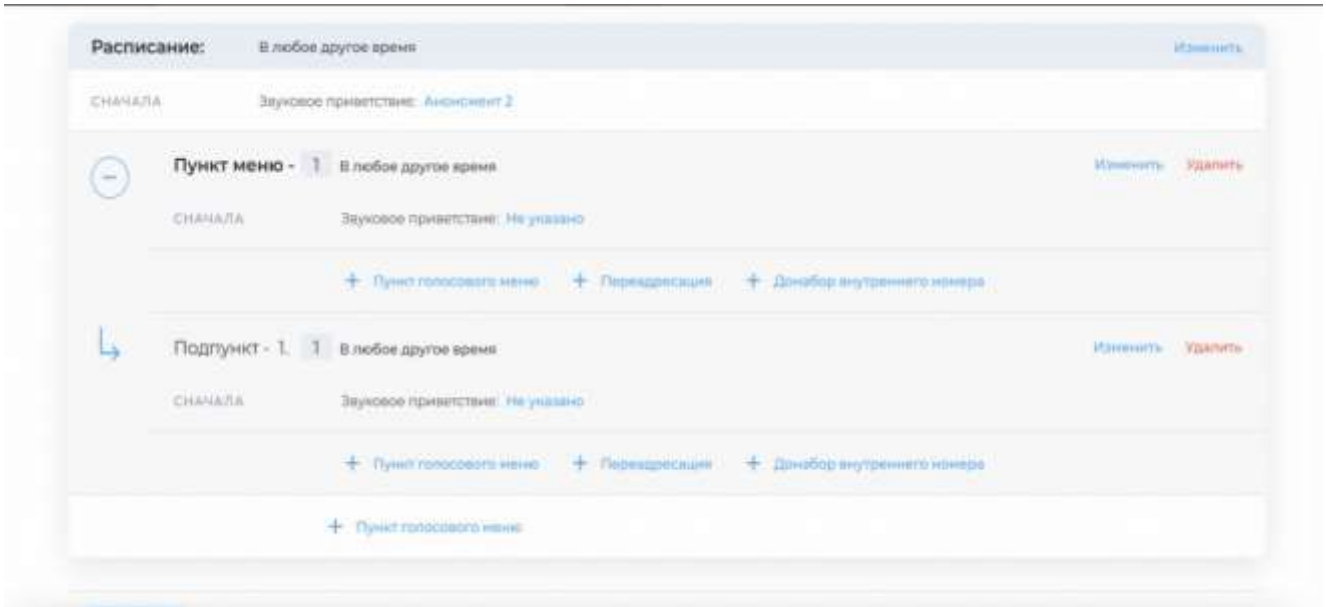

Рисунок 4.22 - Додавання підпункту голосового меню

Дуже зручною є функція розкладу роботи IVR, налаштувати яку дуже просто, якщо дотримуватись наступних кроків:

1. Натиснути на кнопку «Змінити», що знаходиться напроти поля «Розклад» (рис.3);
2. Обрати звукове привітання;
3. Підтвердити вибір звукового привітання, натиснувши на кнопку «Підтвердити»;
4. У відкритому вікні натиснути кнопку «Змінити» (рис.4.23);

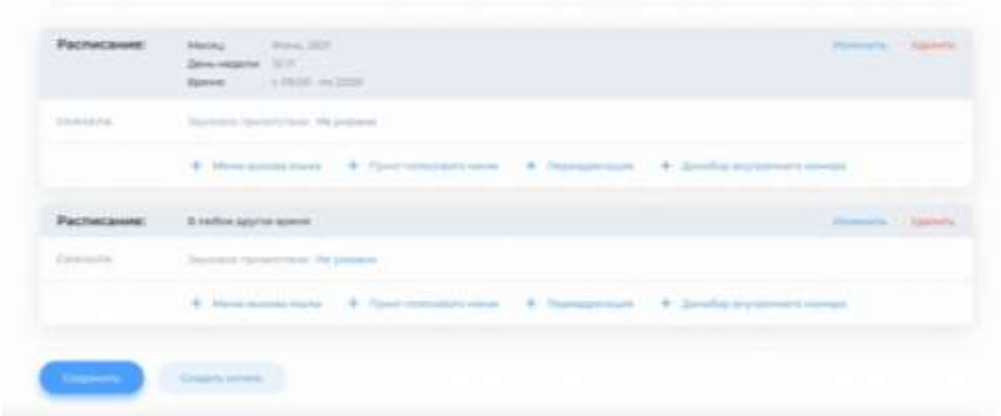

Рисунок 4.23– Редагування розкладу

1. Поставити прапорець біля слова «Період» (рис.4.24);
2. Обрати потрібний місяць для роботи IVR;
3. Обрати перший та останній дні роботи IVR;
4. Вказати бажаний діапазон роботи IVR в годинах(в діапазоні 00:00 – 23:59);
5. Натиснути кнопку «Зберегти».

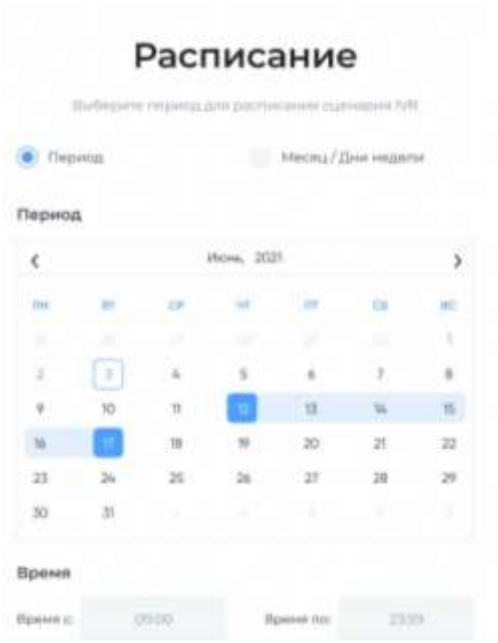

Рисунок 4.24 – Вибір дати роботи IVR

Таким чином, розроблено інструкцію користувача.

4.5 Висновки

Проаналізовано доцільність та актуальність проведення модульного тестування та прийнято рішення виконати модульне тестування найбільш критичних елементів системи.

Проаналізовано доцільність використання запропонованого методу інтерактивного вибору стратегії гри, шляхом моделювання гри двох гравців, один з яких використовує дану стратегію, на випадковому наборі вхідних даних.

Розроблено модульні тести для модуля для синхронізації дій гравця та ходу гри на сервері, модуля для вибору ходу комп’ютером, який в тому числі містить логіку для вибору стратегії гри, модуля для представлення стану карти, модуля для випадкової генерації колоди.

Проаналізовано доцільність та проведено тестування системи методом чорної скриньки.

Розроблено інструкцію користувача.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Програмна система оптимізації створення голосового меню IVR» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Програмна система оптимізації створення голосового меню IVR» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [29].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	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 працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на 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

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	4	3	4
3. Ринкові переваги (ціна продукту)	2	2	2
4. Ринкові переваги (технічні властивості)	5	3	4
5. Ринкові переваги (експлуатаційні витрати)	1	1	1
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	1	1	1
8. Практична здійсненність (наявність фахівців)	3	3	3

9. Практична здійсненність (наявність фінансів)	3	4	4
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	4	3
Сума балів	39	37	38
Середньоарифметична сума балів $СБ_c$	38,0		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 **[**29**]**.

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Програмна система оптимізації створення голосового меню IVR» становить 38,0 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Оцінювання рівня новизни розробки

Виводячи на ринок новинку виробник вважає, що тієї новизни, якою наділена нова розробка є достатньо для того, щоб вона була сприйнята споживачем як нова. Але це не завжди так, в силу того, що споживач і виробник неоднозначно визначать її рівень новизни. Тому доцільним є визначення рівня новизни розробки отриманої в результаті досліджень за темою «Програмна система оптимізації створення голосового меню IVR».

Саме визначення рівня і ступеня інтегральної новизни є найбільш актуальним, оскільки її рівень визначає ступінь однакового позитивного сприйняття новизни розробки як виробником, так і споживачем, а отже і ринком в цілому, а це, у свою чергу, є гарантією того, що новинка знайде своє місце на ринку, користуватиметься попитом у споживачів і забезпечить відшкодування витрат, зазнаних товаровиробником під час розроблення та виробництва технічної розробки [29].

Рівень новизни нової продукції розраховуємо експертним методом шляхом протиставлення нової продукції та її аналогів, що існують в даний час на ринку, за чинниками що визначають її значення, в системі «краще-гірше». Рівень новизни встановлюємо відносно рівня аналога (або продукту, що досить близький до аналога).

Для визначення *i*-го виду новизни, застосуємо чинники, які впливають на її рівень. Кожен чинник *i*-го виду новизни розраховуємо в балах. Більша кількість набраних балів свідчить про більший рівень новизни. Для оцінювання рівня новизни використаємо думки експертів, які встановлюють визначені бали відповідним чинникам. Бал відповідності проставляється в діапазоні від (-5 – значно гірше аналога до +5 – значно краще аналога). Результати попереднього оцінювання зведемо до відповідного листа оцінювання (таблиця 5.4).

Таблиця 5.4 – Лист оцінювання рівня новизни експертами

Види та чинники		Бали та експерти		
		Експерт 1	Експерт 2	Експерт 3
1		*2*	*3*	*4*
Споживча новизна	Питома вага 0,24	Максимальний бал $Б_{i\ max}$		25
1. Зміна поведінкових звичок споживача		4	4	4
2. Ступінь задоволення потреб і запитів		5	5	5
3. Спосіб задоволення потреби		3	3	4
4. Формування нової потреби		3	3	2
5. Формування нового споживача		0	0	0
Середній бал експертів $Б_{i\ отр}$		15		
Товарна новизна	Питома вага 0,202	Максимальний бал $Б_{i\ max}$		30
1. Параметричні зміни показників продукції				

1.1. Якісні		3	4	3
1.2. Технічні		4	4	3
1.3. Економічні		3	3	3
1.4. Сервісні		4	4	4
2. Якість продукції по відношенню до конкурентів		3	3	3
3. Функціональні зміни		3	3	3
Середній бал експертів $Б_{i\ отр}$		20		
Виробнича новизна	Питома вага 0,042	Максимальний бал $Б_{i\ MAX}$		25
1. Рівень унікальності товару для підприємства		5	5	5
2. Рівень унікальності для галузі		3	4	3
3. Рівень унікальності товару для країни		1	1	1
4. Зміна виробничої системи		4	4	4
5. Відносно існуючого асортименту		2	2	2
Середній бал експертів $Б_{i\ отр}$		15		
Прогресивна новизна	Питома вага 0,2	Максимальний бал $Б_{i\ MAX}$		25
1. Зміна технології виготовлення		4	4	4
2. Рівень застосування нових компонентів і матеріалів		1	2	1
3. Зміна технологічного принципу дії виробу		1	2	1
4. Зміна конструктивного виконання		3	2	3
5. Рівень застосування інновацій		2	2	2
Середній бал експертів $Б_{i\ отр}$		11		
Ринкова новизна	Питома вага 0,1	Максимальний бал $Б_{i\ MAX}$		20
1. Новий виріб на новому ринку		0	0	0
2. Новий виріб на відомому ринку		2	2	2
3. Модернізований виріб		2	2	2
4. Нова модель		1	2	2
Середній бал експертів $Б_{i\ отр}$		6		
Екологічна новизна	Питома вага 0,035	Максимальний бал $Б_{i\ MAX}$		20
1. Рівень екологічної чистоти технології виробництва		5	5	5
2. Рівень впровадження мало- та безвідходних технологій		5	5	5
3. Рівень екологічно небезпечних режимів експлуатації продукції		5	5	5
4. Рівень забруднення навколишнього середовища		5	5	5
Середній бал експертів $Б_{i\ отр}$		20		
Соціальна новизна	Питома вага 0,036	Максимальний бал $Б_{i\ MAX}$		20
1. Використання нового товару приводить до покращення стану здоров’я нації		0	0	0
2. Використання нового товару приводить до зростання доходів населення		0	0	0

3. Виробництво нового товару приводить до збільшення (зменшення) кількості робочих місць на підприємстві		4	5	4
4. Виробництво нового товару приводить до підвищення кваліфікації персоналу		3	3	3
Середній бал експертів $Б_{i\ отр}$		7		
Маркетингова новизна	Питома вага 0,145	Максимальний бал $Б_{i\ MAX}$		20
1. Нові методи маркетингових досліджень		0	0	0
2. Вживання нових стратегій сегментації ринку		3	3	3
3. Вибір нової маркетингової стратегії обхвату і розвитку цільового сегмента		2	3	2
4. Побудова нових каналів збуту		0	1	1
Середній бал експертів $Б_{i\ отр}$		6		

Значення *i*-го виду новизни розрахуємо за формулою [30]:

$$I_i = \frac{Б_{i\,отр}}{Б_{i\,MAX}}, \tag{5.1}$$

де $Б_{i\ отр}$ – отримана кількість балів за шкалою оцінок чинників, що визначають *i*-й вид новизни;

$Б_{i\ MAX}$ – максимальна кількість балів, що може бути отримана за *i*-м видом новизни.

Загальний рівень інтегральної новизни розраховуємо шляхом перемноження отриманого значення *i*-го виду новизни на її вагомість, причому вагомість *i*-го виду новизни визначаємо експертним методом, за формулою [30]:

$$N_{інт} = \sum_{i}^{n} W_i \cdot I_i, \tag{5.2}$$

де $N_{інт}$ – рівень інтегральної (сукупної) новизни;

W_i – вагомість (питома вага) *i*-го виду новизни;

n – загальна кількість видів новизни.

$$N_{інт} = (0{,}24 \cdot 15/25)+(0{,}202 \cdot 20/30)+(0{,}042 \cdot 15/25)+(0{,}2 \cdot 11/25)+(0{,}1 \cdot 6/20)+$$
$$(0{,}035 \cdot 20/20)+(0{,}036 \cdot 7/20)+(0{,}145 \cdot 6/20) = 0{,}515.$$

Отримане значення інтегрального рівня новизни зіставляємо зі шкалою, що наведена в табл. 5.5 [30].

Таблиця 5.5 – Рівні новизни нового товару та їхня характеристика

Рівні новизни товару	Значення інтегральної новизни	Характеристика товару	Вид нового товару
Найвища	1,00	Абсолютно новий товар	Новий товар, що наділений ознаками інноваційності (інноваційний товар)
Висока	0,8…0,99	Товар, який не має аналогів	
Значуща	0,6…0,79	Принципова зміна споживчих властивостей товару	
Достатня	0,4…0,59	Принципова технологічна модифікація товару	
Незначна	0,2…0,39	Кардинальна зміна параметрів	Новий товар
Помилкова	0,00…0,19	Малоістотна модифікація	

Згідно таблиці 5.5 розробка відповідає рівню при значенні інтегральної новизни 0,515 - достатня новизна; за характеристикою: принципова технологічна модифікація товару; вид розробки - новий товар, що наділений ознаками інноваційності (інноваційний товар).

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Програмна система оптимізації створення голосового меню IVR», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [30]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.3)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=24 дні.

$$З_о = 12820{,}00 \cdot 24 / 24 = 12820{,}00 \; грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	12820,00	534,17	24	12820,00
Інженер-розробник програмного забезпечення	12345,00	514,38	20	10287,50
Інженер-системотехнік (побудова автоматизованих і програмних систем)	12100,00	504,17	10	5041,67
Лаборант	6950,00	289,58	10	2895,83
Всього				31045,00

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Програмна система оптимізації створення голосового меню IVR» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (5.4)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.5)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=2379,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [30];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 24 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 2379{,}00 \cdot 1{,}10 \cdot 1{,}65 / (24 \cdot 8) = 22{,}49$ *грн.*
$З_{р1} = 22{,}49 \cdot 8{,}20 = 184{,}41$ *грн.*

Таблиця 5.7 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка електронно-обчислювального обладнання	8,20	2	1,10	22,49	184,41
Підготовка робочого місця розробника програмного забезпечення	6,50	2	1,10	22,49	146,18
Інсталяція програмного забезпечення голосового аналізатора	5,50	5	1,70	34,76	191,16
Монтаж дослідної серверної частини	6,50	3	1,35	27,60	179,40

Введення кодів програмних модулів	13,00	3	1,35	27,60	358,80
Підготовка програмного забезпечення	8,00	5	1,70	34,76	278,05
Формування бази даних голосових команд	12,00	2	1,10	22,49	269,87
Тестування програмного забезпечення	6,00	2	1,10	22,49	134,93
Всього					1742,79

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.6)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (31045,00 + 1742,79) · 11 / 100% = 3606,66 грн.

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.7)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (31045,00 + 1742,79 + 3606,66) · 22 / 100% = 8006,78 грн.

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Програмна система оптимізації створення голосового меню IVR».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \quad (5.8)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, (K_j = *1,1 … 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

M_1 = *4,00 · 162,00 · 1,1 - 0,000 · 0,00 = 712,80 грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір 500-80 Crystal	162,00	4,00	0,000	0,00	712,80
Папір для записів А5 DATA	75,00	4,00	0,000	0,00	330,00
Органайзер офісний Office	310,00	3,00	0,000	0,00	1023,00
Канцелярське приладдя (набір офісного працівника)	162,00	3,00	0,000	0,00	534,60
Картридж для принтера Canon LBP	842,00	1,00	0,000	0,00	926,20
Диск оптичний OpticVisio CD-RW	13,50	3,00	0,000	0,00	44,55
Flesh-пам'ять DATA 64 GB	320,00	2,00	0,000	0,00	704,00
Тека для паперів	102,00	3,00	0,000	0,00	336,60
Інше	110,00	1,0	0,000	0,00	121,00
Всього					4732,75

5.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Програмна система оптимізації створення голосового меню IVR» відсутні.

5.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (5.9)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 27000,00 \cdot 1 \cdot 1,11 = 29970,00$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.9 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Серверне обладнання розміщення програмного забезпечення CUBE XT-6548 A341	1	27000,00	29970,00
Обладнання передачі даних	1	7605,00	8441,55
Всього			38411,55

5.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

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

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

k – кількість найменувань програмних засобів.

$B_{прг} = 12670,00 \cdot 1 \cdot 1,1 = 13937,00$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.10 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Програмне забезпечення обробки голосових даних	1	12670,00	13937,00
Програмне забезпечення підтримки серверного обладнання	1	9850,00	10835,00
Всього			24772,00

5.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} , \qquad (5.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (29385{,}00 \cdot 1) / (2 \cdot 12) = 1224{,}38$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.11 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер	29385,00	2	1	1224,38
Робоче місце розробника	8645,00	5	1	144,08
Пристрої виводу інформації	6200,00	4	1	129,17
Оргтехніка	6890,00	4	1	143,54
Приміщення лабораторії	245000,00	25	1	816,67
ОС Windows 11	7352,00	2	1	306,33
Прикладний пакет Microsoft Office 2019	6542,00	2	1	272,58
Засоби передачі даних	6750,00	2	1	281,25
Всього				3318,00

5.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{впі}}{\eta_i}, \qquad (5.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е$ = 4,50 грн;

$K_{вп i}$ – коефіцієнт, що враховує використання потужності, $K_{вп i} < 1$;

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

$B_е = 0,65 \cdot 164,0 \cdot 4,50 \cdot 0,95 / 0,97 = 479,70$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.12 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер	0,65	164,0	479,70
Робоче місце розробника	0,20	162,0	145,80
Пристрої виводу інформації	0,02	20,0	1,80
Оргтехніка	0,80	12,0	43,20
Засоби передачі даних	0,05	135,0	30,38
Серверне обладнання розміщення програмного забезпечення CUBE XT-6548 A341	0,55	135,0	334,13
Обладнання передачі даних	0,07	135,0	42,53
Всього			1077,53

5.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Програмна система оптимізації створення голосового меню IVR» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.13)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (31045,00 + 1742,79) · 20 / 100% = 6557,56 грн.

5.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.14)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (31045,00 + 1742,79) · 30 / 100% = 9836,34 грн.

5.3.11 Інші 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» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 60%.

$І_в$ = (31045,00 + 1742,79) · 60 / 100% = 19672,68 грн.

5.3.12 Накладні (загальновиробничі) витрати

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» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 125%.

$B_{нзв}$ = (31045,00 + 1742,79) · 125 / 100% = 40984,74 грн.

Витрати на проведення науково-дослідної роботи на тему «Програмна система оптимізації створення голосового меню IVR» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (5.17)$$

$B_{заг}$= 31045,00 +1742,79 +3606,66 +8006,779311 +4732,75 +0,00 +38411,55 + 24772,00 + 3318,00 +1077,53 +6557,56 +9836,34 +19672,68 +40984,74 = 193764,37 грн.

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

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.18)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 193764,37 / 0,95 = 203962,50 грн.

5.3.13 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим 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.

Результати дослідження проведені за темою «Програмна система оптимізації створення голосового меню IVR» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	2500	5000	5500	4250

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 24000 осіб;

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 180,00 грн;

$\pm\Delta Ц_o$ – зміна вартості послуги від впровадження результатів, приймемо 10,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [30]**:**

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.19)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 42\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (10,00 \cdot 24000,00 + 190,00 \cdot 2500) \cdot 0,83 \cdot 0,42 \cdot (1 - 0,18/100\%) = 204384,18$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (10,00 \cdot 24000,00 + 190,00 \cdot 7500) \cdot 0,83 \cdot 0,42 \cdot (1 - 0,18/100\%) = 475943,58$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (10,00 \cdot 24000,00 + 190,00 \cdot 13000) \cdot 0,83 \cdot 0,42 \cdot (1 - 0,18/100\%) = 774658,92$ грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (10,00 \cdot 24000,00 + 190,00 \cdot 17250) \cdot 0,83 \cdot 0,42 \cdot (1 - 0,18/100\%) = 1005484,41$ грн.

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

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{5.20}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,14$;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =204384,18/(1+0,14)[1]+475943,58/(1+0,14)[2]+774658,92/(1+0,14)[3]+
+1005484,41/(1+0,14)[4]=179284,37+366223,13+522872,71+595327,49=1663707,69 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{5.21}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=1,5;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 203962,50 грн.

$PV = k_{інв} \cdot ЗВ$ = 1,5 · 203962,50 = 305943,74 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.22}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 1663707,69 грн;

PV – теперішня вартість початкових інвестицій, 305943,74 грн.

$E_{абс} = ПП - PV$ = 1663707,69 - 305943,74 = 1357763,95 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1+\frac{E_{абс}}{PV}} - 1, \quad (5.23)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 1357763,95 грн;

PV – теперішня вартість початкових інвестицій, 305943,74 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Tж]{1+\frac{E_{абс}}{PV}} - 1 = (1+1357763,95/305943,74)^{1/4} = 0,53.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.24)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,16.

$\tau_{мін}$ = 0,11+0,16 = 0,27 < 0,53 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Програмна система оптимізації створення голосового меню IVR» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (5.25)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,53 = 1,90 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Програмна система оптимізації створення голосового меню IVR» становить 38,0 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 1,90 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Програмна система оптимізації створення голосового меню IVR».

ВИСНОВКИ

У магістерській кваліфікаційній роботі був розроблений метод для покращення процесу створення голосового меню IVR.

Виконано аналіз основних аналогів, виявлено їхні переваги та недоліки. На основі отриманих результатів розроблено веб-інтерактивний додаток.

Було розглянуто існуючі аналоги та їх недоліки, зроблено порівняння з власним програмним продуктом та визначено, що є доцільним розробити новий програмний додаток цього типу. Виконано постановку задач для розробки.

У результаті проведення варіантного аналізу, було обрано мову програмування PHP та офіційну IDE –PhpStorm.

Були вирішенні такі задачі:

- проведено аналіз аналогів;
- проведено аналіз засобів та методів розробки IVR;
- розроблено власний покращений алгоритм створення меню IVR;
- розроблено алгоритм роботи модуля, що відповідає за синхронізацію з сервером asterisk;
- розроблено API для GrapQl;
- розроблено клієнтський застосунок з використанням запропонованого алгоритму;
- проведено тестування системи.

Розроблено загальну модель системи та алгоритм покращеного створення голосового меню IVR. В процесі розробки реалізовано інтерфейс додатку та основні модулі.

Магістерську кваліфікаційну роботу було оформлено відповідно до вимог.

Тестування програми довело повну працездатність програмного продукту та відповідність поставленому технічному завданню. Розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лічман Едуард Русланович Розробка інтерактивних веб додатків // XLVIII Молодіжна наукова ліга (2020) [Електронний ресурс] – Режим доступу до до ресурсу: https://ojs.ukrlogos.in.ua/index.php/liga/issue/view/29.05.2020/309

2. Sierra K. Head First Java / K. Sierra, B. Bates., 2009. – O'Reilly, – 254 с

3. Android Studio Development (2017) [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/

4. Stack Overflow [Електронний ресурс] // Stack Exchange Inc. – 2017 Режим доступу до ресурсу: https://ru.stackoverflow.com/.

5. Delivery Club [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Delivery_Club

6. Якитория [Електронний ресурс] – Режим доступу до ресурсу: https://yaki.kh.ua/

7. FoodPanda [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Foodpanda

8. Інформаційні системи і технології в статистиці: Навч. посібник / За ред. д-ра екон. наук, проф. В. Ф. Ситника. — К.: КНЕУ, 2003. — 267 с.

9. Barry, Burd Android Application Development All–in–One For Dummies® / Barry Burd. - Москва: Машиностроение, 2011. - 816 с.

10. Биллиг, В. А. Основы объектного программирования на C# (C# 3.0, Visual Studio 2008) / В.А. Биллиг. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2010. - 584 с.

11. Гарнаев, Андрей WEB-программирование на Java и JavaScript / Андрей Гарнаев , Сергей Гарнаев. - М.: БХВ-Петербург, 2012. - 179 с.

12. Voxium [Електронний ресурс] – Режим доступу до ресурсу: https://voximplant.ru/solutions/smart-ivr

13. Давыдов, Станислав PhpStrorm. Профессиональное программирование на php. Наиболее полное руководство (+ CD-ROM) / Станислав Давыдов , Алексей Ефимов. - М.: БХВ-Петербург, 2011. - 800 с.

14. Дэрси, Лорен Android за 24 часа. Программирование приложений под операционную систему Google / Лорен Дэрси , Шейн Кондер. - М.: Рид Групп, 2011. - 464 с.

15. Майер, Рето Android 2. Программирование приложений для планшетных компьютеров и смартфонов / Р. Майер. - М.: "Издательство "Эксмо", 2011.- 672 с.

16. Майер, Рето Android 4. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер. - М.: Эксмо, 2013. - 816 с.

17. Мартин, К. Соломон Oracle. Программирование на языке Java / Мартин К. Солом, Нирва Морисо-Леруа , Джули Басу. - М.: ЛОРИ, 2010. - 512 с.

18. Машнин, Т. С. Eclipse. Разработка RCP-, Web-, Ajax- и Android-приложений на Java / Т.С. Машнин. - М.: БХВ-Петербург, 2013. - 384 с.

19. Нотон Php. Справочное руководство. Все, что необходимо для программирования на Php / Нотон, Патрик. - М.: Бином, 2015. - 448 с.

20. Осипов, Дмитрий Delphi. Программирование для php, Дмитрий Осипов. - М.: "БХВ-Петербург", 2014. - 464 с.

21. Роджерс, Рик Android. Разработка приложений / Рик Роджерс и др. - М.: ЭКОМ Паблишерз, 2010. - 400 с.

22. Положення про кваліфікаційну роботу у Вінницькому національному технічному університеті / Уклад. О. Н. Романюк, Р. Р. Обертюх, Т. О. Савчук, Л. П. Громова - Вінниця : ВНТУ, 2015 - 27 с.

23. Bishop G. F., Weimer D. M. FastPhongShading / G. F. Bishop, D. M. Weimer // ComputerGraphics, v20, n4. – 1986.

24. MontesRosana, UrenaCarlos. AnOverviewof BRDF Models / RosanaMontes, CarlosUrena // TechnicalReport LSI. – 2012.

25. Simon’sTechBlog: Microfacet BRDF. [Електронний ресурс]. – Режим доступу: http://simonstechblog.blogspot.com/2011/12/microfacet-brdf.html

26. WaltdisneyAnimationStudio. [Електронний ресурс]. – Режим доступу: http://www.disneyanimation.com/technology/brdf.html.

27. Аммерал А. Unit test: Пер. с англ. - М.: Солсистем, 1992. - 224 с.

28. Електронний ресурс, режим доступу: http://uareferat.com/Види_тестів_ та_форми_тестових_завдань.

29. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

30. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

ДОДАТОК А.
ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

« 13 » вересня 2021 р.

Технічне завдання

на магістерську кваліфікаційну роботу

« Програмна система оптимізації створення голосового меню IVR »

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

Рейда О. М., к.т.н., доцент кафедри ПЗ

"____"_____2021 р.

Виконав:

студент гр.2ПІ-20м Лічман Е. Р.

"____"_____2021 р.

Вінниця – 2021 рік

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Програмна система оптимізації створення голосового меню IVR».

Галузь застосування – call-центри.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ ректора по ВНТУ № 277 від « 24 » вересня 2021 р. про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності створення інтерактивного голосового меню за рахунок оптимізації передачі даних і використання системних ресурсів.

Призначення роботи – розробка алгоритму створення голосового меню IVR.

3 Вихідні дані для проведення МКР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР:

1. Эдриен М. Использование Docker / Моуэт Эдриен. – Москва, 2017.

2. Adel F. Архитектура сложных веб приложений [Електронний ресурс] / Adel. – 2020.

3. Видеозаписи всех докладов PHP [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/badoo/blog487264/

4. Технічні вимоги

Стандартизоване голосове меню IVR, параметри сценарію IVR, звукові файли із оголошенням пунктів меню IVR.

5. Конструктивні вимоги.

Розробка повинна відповідати усім функціональним вимогам. Повинна бути зручною в використанні та налагодженні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;
– технічне завдання;
– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації та ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану питання та постановка задач дослідження	15.09.2021-26.09.2021
2	Розробка структури програмного продукту	27.09.2021-15.10.2021
3	Розробка програмних засобів	16.10.2021-7.11.2021
4	Тестування системи	8.11.2021-21.11.2021
5.	Економічна частина	22.11.2021-30.11.2021

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б.
ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

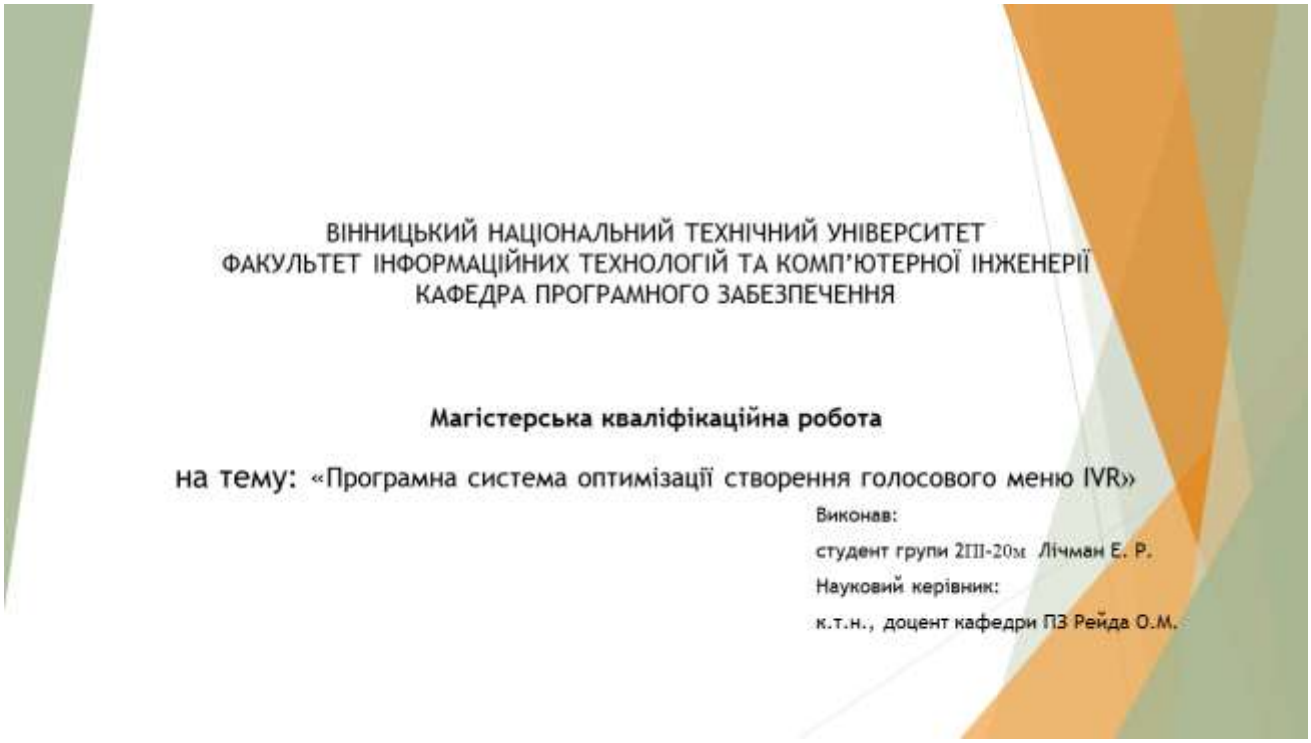

Рисунок Б.1. Плакат №1

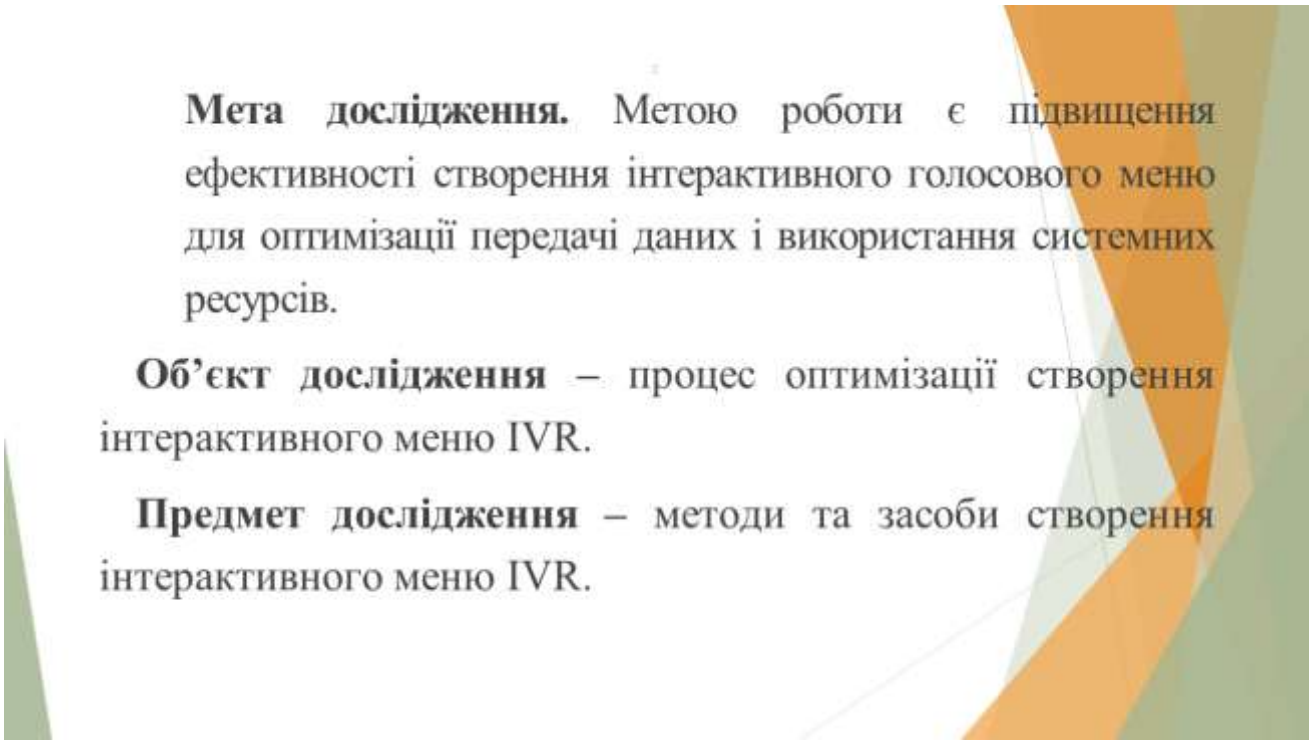

Рисунок Б.2. Плакат №2

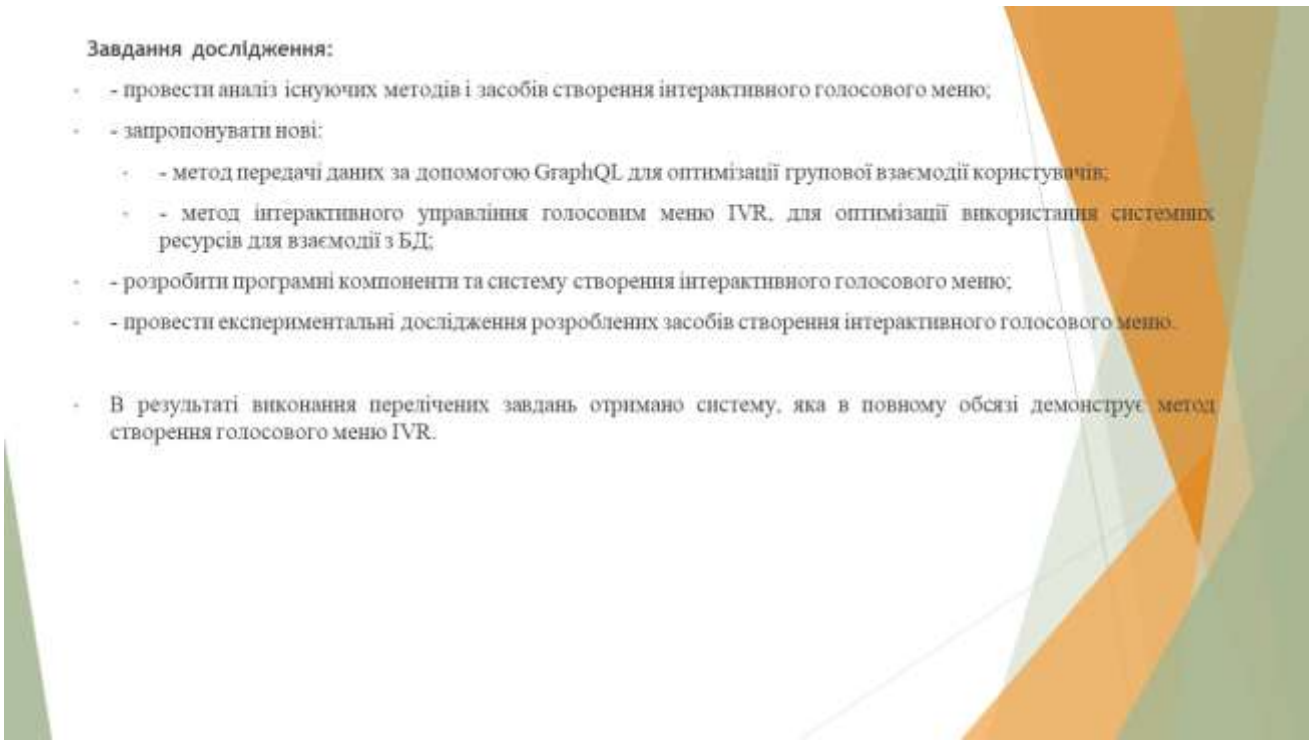

Рисунок Б.3. Плакат №3

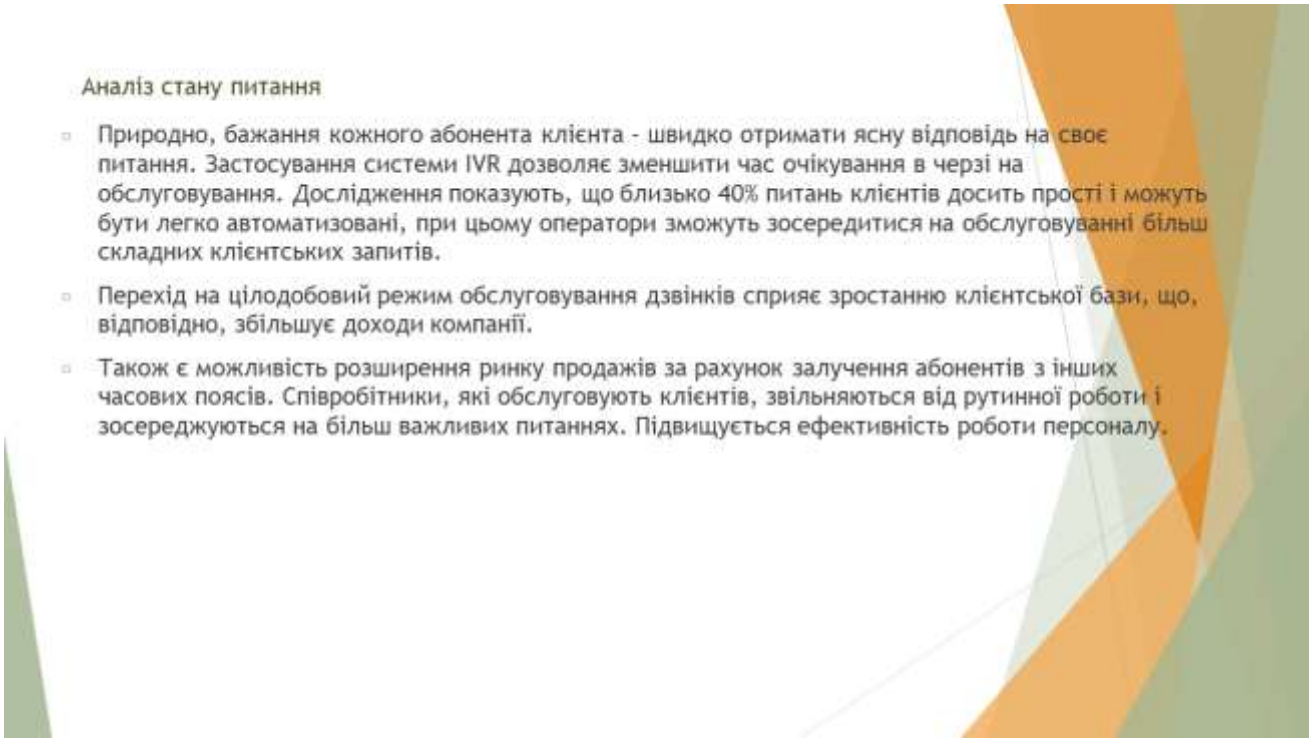

Рисунок Б.4. Плакат №4

Рисунок Б.5. Плакат №5

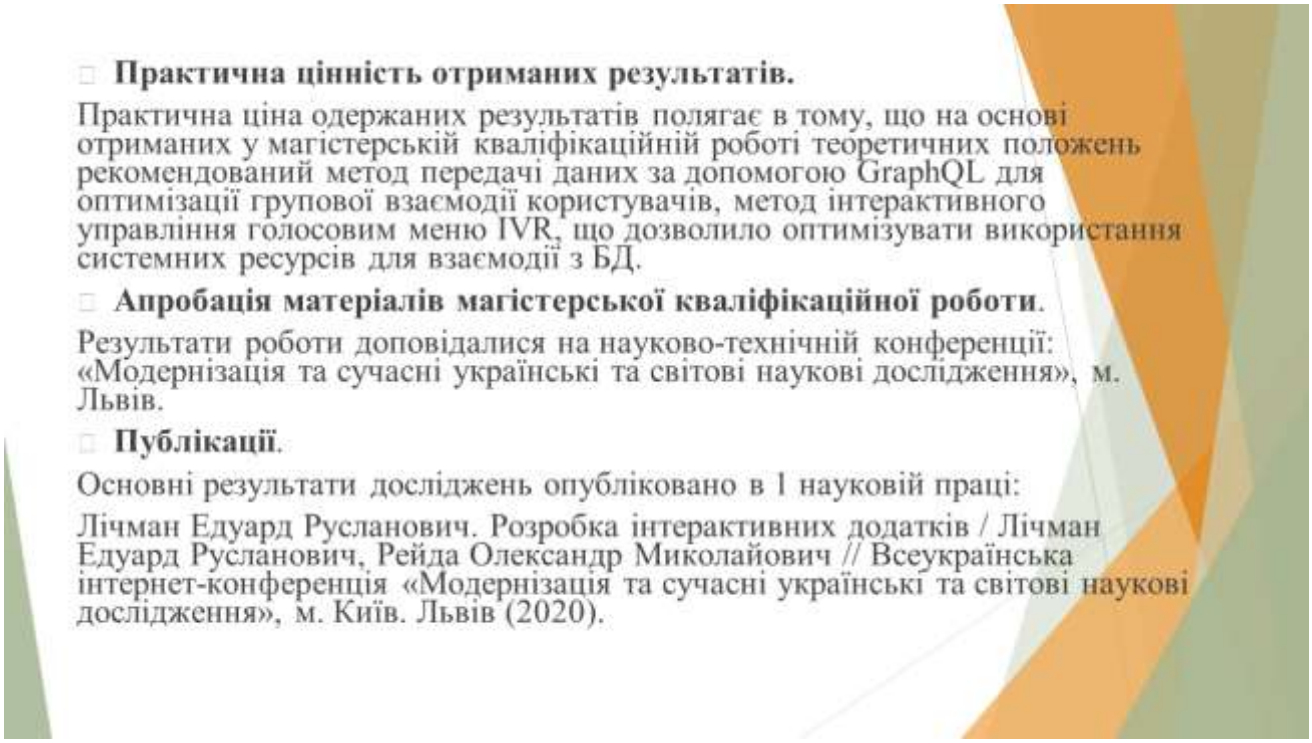

Рисунок Б.6. Плакат №6

Порівняння аналогів

Критерій	zadarma	aimylogic	aimylogic	Власна розробка
Створення голосового меню з власних аудіофайлів	1	1	0	1
Озвучування тексту на різних мовах	0	1	1	1
Вибір голосу озвучування	1	0	1	1
Обробка отриманої від користувача інформації	0	1	1	1
Безкоштовне використання	1	0	0	1
Запис розмови з абонентом	1	0	1	1
Створення шаблонних повідомлень	1	0	0	1
Сума	5	3	4	7

Рисунок Б.7. Плакат №7

Архітектура системи

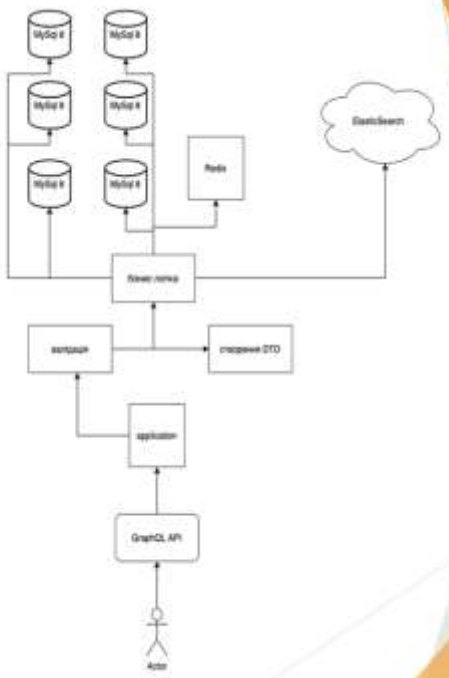

Рисунок Б.8. Плакат №8

Рисунок Б.9. Плакат №9

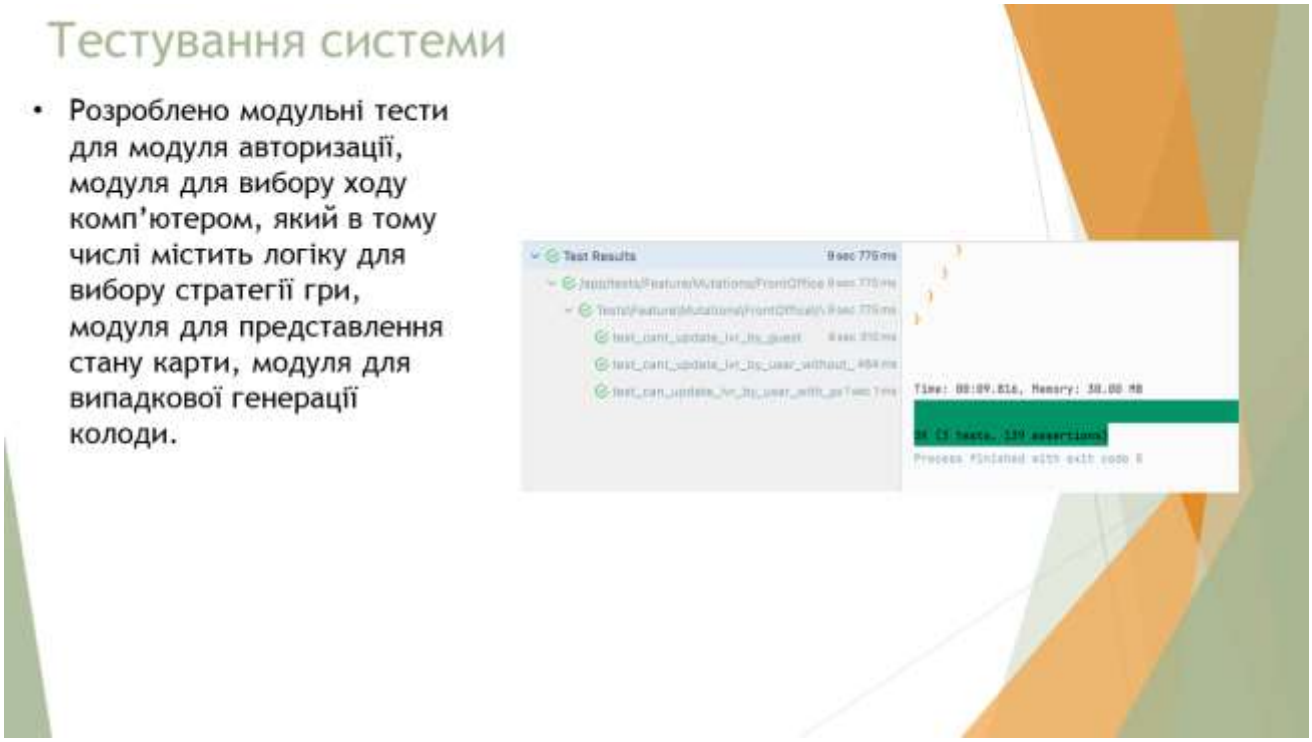

Рисунок Б.10. Плакат №10

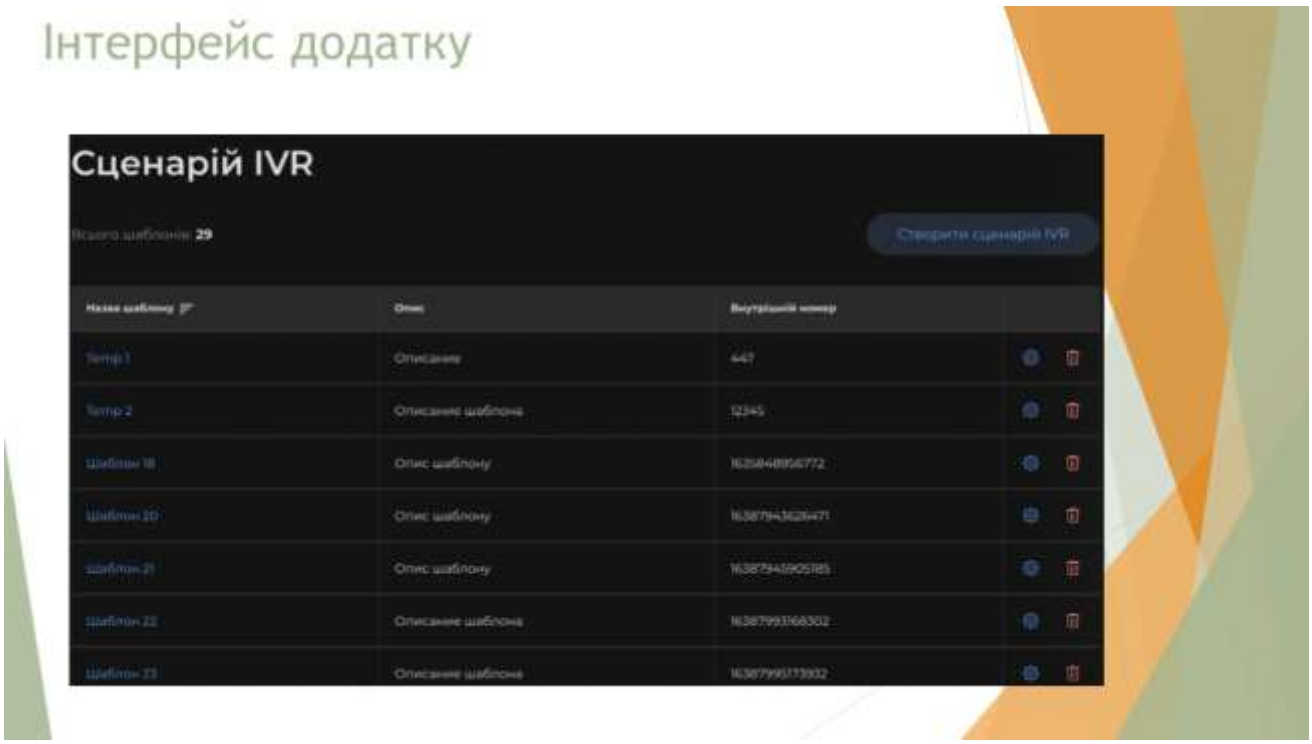

Рисунок Б.11. Плакат №11

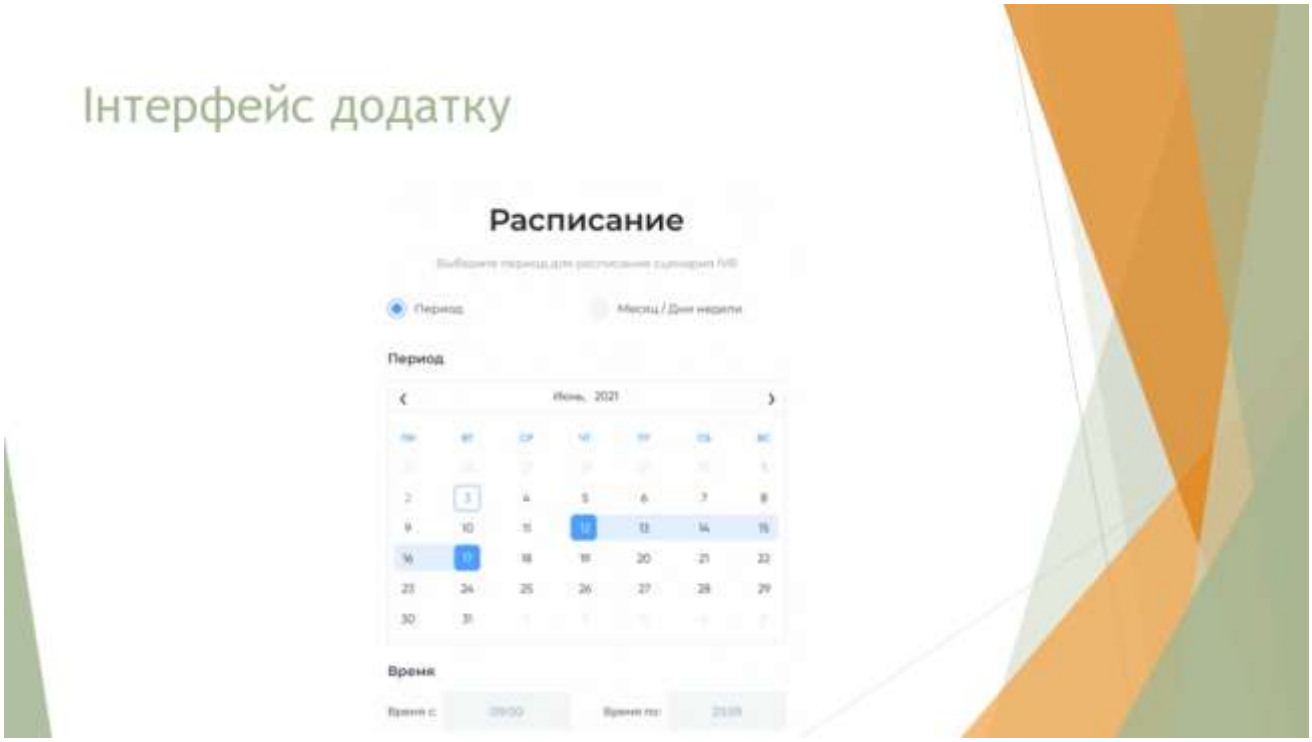

Рисунок Б.12. Плакат №12

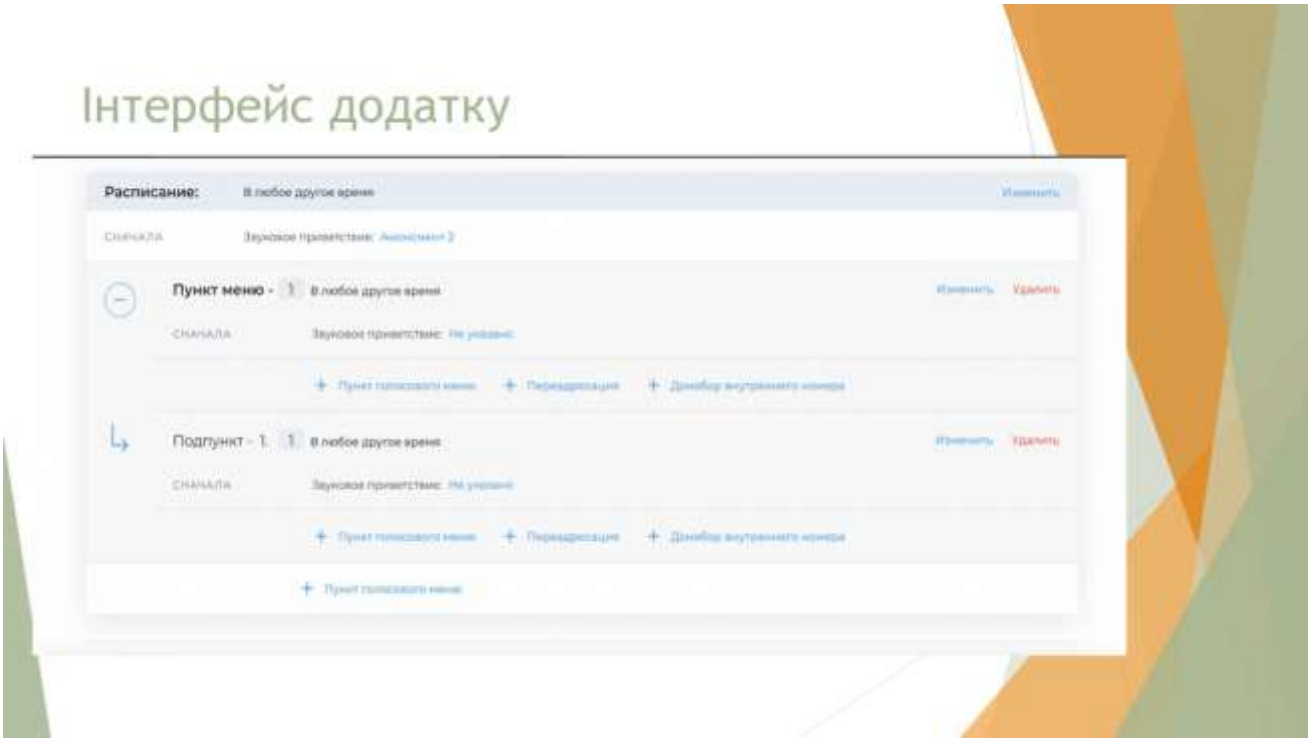

Рисунок Б.13. Плакат №13

Рисунок Б.14. Плакат №14

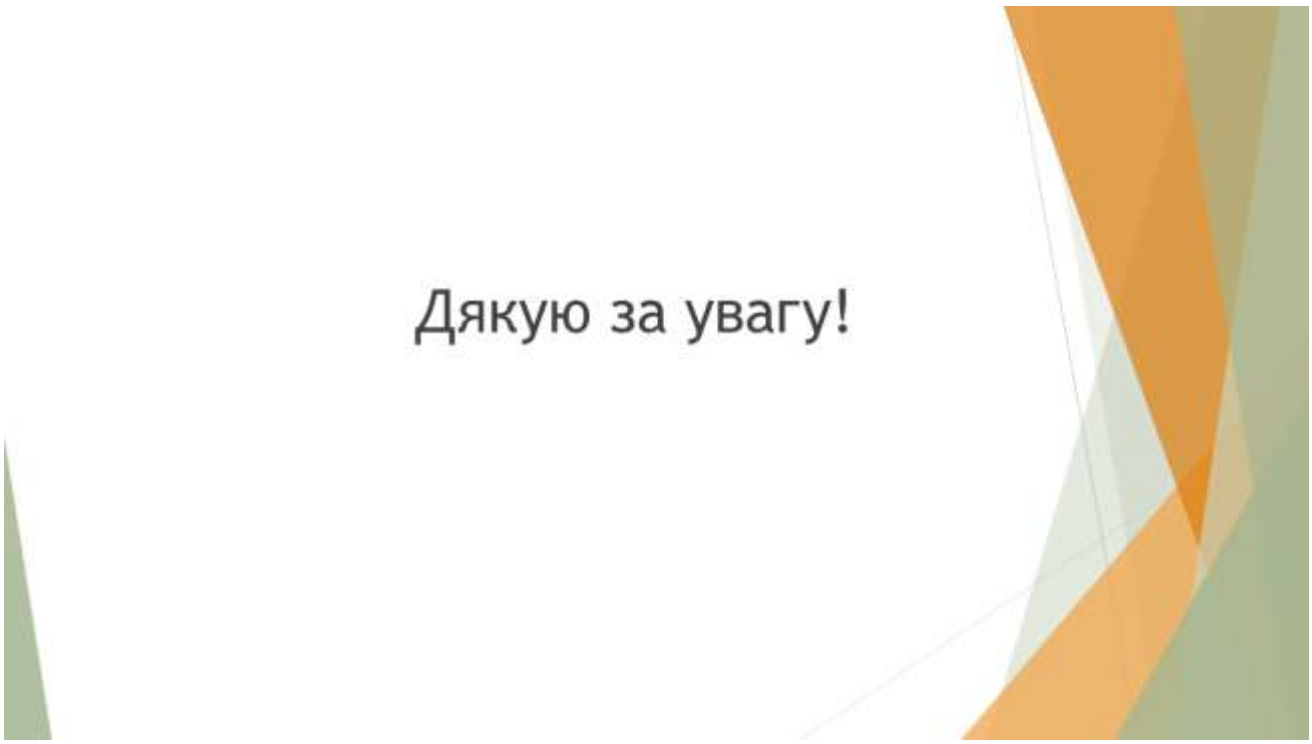

Рисунок Б.15. Плакат №15

ДОДАТОК В.

ЛІСТИНГ МУТАЦІЇ ДЛЯ СТВОРЕННЯ IVR

```
<?php

namespace App\GraphQL\Mutations\FrontOffice\IVR;

use App\Dto\IVR\IvrDto;
use App\Exceptions\TranslatedException;
use App\GraphQL\Types\LocalTypes;
use App\Models\Extensions\IVR;
use App\Permissions\IVR\IVRCreatePermission;
use GraphQL\Type\Definition\ResolveInfo;
use Rebing\GraphQL\Support\SelectFields;
use Throwable;

class IVRCreateMutation extends IVRBaseMutation
{
    public const NAME = 'IVRCreateMutation';
    public const DESCRIPTION = 'Создает IVR.';
    public const PERMISSION = IVRCreatePermission::KEY;

    public function args(): array
    {
        return [
            'name' => LocalTypes::notNullString(),
            'description' => LocalTypes::notNullString(),
        ];
    }

    /**
     * @throws Throwable
     */
    public function doResolve(
        $root,
        array $args,
        $context,
        ResolveInfo $info,
        SelectFields $fields
    ): IVR {
        $company = $this->company();

        if (IVR::query()->whereCompany($company)->count() >=
config('IVR.limit_IVRs_for_company')) {
```

```
            throw new TranslatedException(__('validation.IVR.company_IVRs_limit'));
        }

        return makeTransaction(
            fn() => $this->service->create(IvrDto::byArgs($args), $company),
            $this->service->getConnectionsForTransaction()
        );
    }

    protected function rules(array $args = []): array
    {
        return $this->guest()
            ? []
            : [
                'name' => ['required', 'string'],
                'description' => ['required', 'string'],
            ];
    }
}
```

ДОДАТОК Г.

ЛІСТИНГ СЕРВІСУ РОБОТИ З IVR

```
<?php

namespace App\Services\IVR;

use App\Dto\IVR\IVRAnnouncementDto;
use App\Dto\IVR\IvrDto;
use App\Dto\IVR\IVRLocaleDto;
use App\Dto\IVR\IVRUpdateDTO;
use App\Enums\Audio\AnnouncementTypeEnum;
use App\Enums\IVR\IVRDefaultValuesEnum;
use App\Helpers\DbConnections;
use App\Models\AsteriskCfg\Extension;
use App\Models\AsteriskCfg\IVR as AsteriskIVRSchedule;
use App\Models\AsteriskCfg\IVRLanguage;
use App\Models\AsteriskCfg\IVROption as AsteriskIVROption;
use App\Models\Companies\Company;
use App\Models\Extensions\Domain;
use App\Models\Extensions\IVR;
use App\Models\Extensions\IVRAnnouncement;
use App\Models\Extensions\IVRLocale;
use App\Models\Extensions\IVROption;
use App\Models\Extensions\IVRSchedule;
use App\Models\Extensions\IVRSipEntities;
use App\Services\Companies\CompanyCleanUp;
use App\Services\Companies\CompanyNumberService;
use Illuminate\Database\Eloquent\Collection;
use Throwable;

class IVRService extends BaseIVRService implements CompanyCleanUp
{
    public function __construct(
        private CompanyNumberService $purchasedService,
    ) {
    }

    public function create(IvrDto $dto, Company $company): IVR
    {
        $domain = $company->domains()->firstOrFail();

        $extension = Extension::firstOrCreate(
            [
```

```
            'domain_id' => $domain->getKamailioId(),
            'exten_number' => $dto->getIVRNumber(),
            'exten_alias' => $dto->getIVRNumber(),
        ],
        [
            'app_id' => null,
            'app_type_queue' => null,
            'app_type_followgroup' => null,
            'app_type_ivr' => 1,
            'forward' => null,
        ]
    );

    return IVR::firstOrCreate(
        [
            'name' => $dto->getName(),
            'description' => $dto->getDescription(),
            'extension_id' => $extension->id,
            'ivr_number' => $dto->getIVRNumber(),
            'domain_id' => $domain->id,
            'attempts' => $dto->getAttempts(),
            'timeout' => $dto->getTimeout(),
            'record' => $dto->isRecord(),
            'language_default' => $dto->getLanguageDefault(),
            'language_change' => $dto->isLanguageChange(),
        ]
    );
}

/**
 * @throws Throwable
 */
public function update(IVRUpdateDTO $updateDTO): IVR
{
    $ivrDto = $updateDTO->getIVRDTO();

    $IVRFromDB = IVR::query()
        ->where('id', $ivrDto->getId())
        ->with('schedules')
        ->firstOrFail();

    $domain = Domain::query()
        ->with('company:id,kamailio_id')
        ->findOrFail($ivrDto->getDomainId());
```

```
        $announcementDTO = $updateDTO->getAnnouncementDTO();
        $languageDefault = $this->getDefaultLanguage($updateDTO);

        $this->updateExtensionInAsterisk($IVRFromDB, $updateDTO, $domain);

        $IVRFromDB->update(
            [
                'name' => $ivrDto->getName(),
                'description' => $ivrDto->getDescription(),
                'ivr_number' => $ivrDto->getIVRNumber(),
                'domain_id' => $domain->id,
                'attempts' => $ivrDto->getAttempts(),
                'timeout' => $ivrDto->getTimeout(),
                'return_code' => $ivrDto->getReturnCode(),
                'record' => $ivrDto->isRecord(),
                'language_default' => $languageDefault,
                'language_change' => $ivrDto->isLanguageChange(),
            ]
        );

        $this->updateOrCreateIVRAnnouncements($IVRFromDB->id,
$announcementDTO);
        $this->updateOrCreateSipEntities($IVRFromDB->id, $updateDTO);
        $this->createIVRLocales($IVRFromDB->id, $updateDTO);

        $this->updateRelatedFieldsInAsteriskForIVR($IVRFromDB, $updateDTO,
$domain);

        return $IVRFromDB;
    }

    private function getDefaultLanguage(IVRUpdateDTO $updateDTO): string
    {
        $languageDefault = IVRDefaultValuesEnum::DEFAULT_LANGUAGE_FIELD;

        foreach ($updateDTO->getLocaleDTO() as $value) {
            if ($value->isMain()) {
                $languageDefault = $value->getLanguage();
            }
        }

        return $languageDefault;
    }
```

```
    protected function updateExtensionInAsterisk(IVR $IVRFromDB, IVRUpdateDTO
$updateDTO, Domain $domain): void
    {
        $ivrDto = $updateDTO->getIVRDTO();
        Extension::query()
            ->where('id', $IVRFromDB->extension_id)
            ->update(
                [
                    'domain_id' => $domain->getKamailioId(),
                    'exten_number' => $ivrDto->getIVRNumber(),
                    'exten_alias' => $ivrDto->getIVRNumber(),
                ]
            );
    }

    private function updateOrCreateIVRAnnouncements(int $IVRId,
IVRAnnouncementDto $announcementDTO): void
    {
        IVRAnnouncement::query()->updateOrCreate(
            [
                'announced_id' => $IVRId,
                'announced_type' => IVR::MORPH_NAME,
                'field' => AnnouncementTypeEnum::TIMEOUT,
            ],
            [
                'announcement_id' => $announcementDTO->getTimeoutAnnouncement(),
            ]
        );

        IVRAnnouncement::query()->updateOrCreate(
            [
                'announced_id' => $IVRId,
                'announced_type' => IVR::MORPH_NAME,
                'field' => AnnouncementTypeEnum::LANGUAGE,
            ],
            [
                'announcement_id' => $announcementDTO->getLanguageAnnouncement(),
            ]
        );

        IVRAnnouncement::query()->updateOrCreate(
            [
                'announced_id' => $IVRId,
```

```
            'announced_type' => IVR::MORPH_NAME,
            'field' => AnnouncementTypeEnum::SCHEDULE,
        ],
        [
            'announcement_id' => $announcementDTO->getLanguageAnnouncement(),
        ]
    );

    IVRAnnouncement::query()->updateOrCreate(
        [
            'announced_id' => $IVRId,
            'announced_type' => IVR::MORPH_NAME,
            'field' => AnnouncementTypeEnum::INVALID,
        ],
        [
            'announcement_id' => $announcementDTO->getInvalidAnnouncement(),
        ]
    );

    IVRAnnouncement::query()->updateOrCreate(
        [
            'announced_id' => $IVRId,
            'announced_type' => IVR::MORPH_NAME,
            'field' => AnnouncementTypeEnum::ATTEMPTS,
        ],
        [
            'announcement_id' => $announcementDTO->getAttemptsAnnouncement(),
        ]
    );

    IVRAnnouncement::query()->updateOrCreate(
        [
            'announced_id' => $IVRId,
            'announced_type' => IVR::MORPH_NAME,
            'field' => AnnouncementTypeEnum::EXTDIAL,
        ],
        [
            'announcement_id' => $announcementDTO->getExtdialAnnouncement(),
        ]
    );

    IVRAnnouncement::query()->updateOrCreate(
        [
            'announced_id' => $IVRId,
```

```
            'announced_type' => IVR::MORPH_NAME,
            'field' => AnnouncementTypeEnum::EXTDIAL_INVALID,
        ],
        [
            'announcement_id' => $announcementDTO-
>getExtdialInvalidAnnouncement(),
        ]
    );
  }

  private function updateOrCreateSipEntities(int $IVRId, IVRUpdateDTO
$updateDTO): void
  {
    $sipEntityDto = $updateDTO->getSipEntityDTO();
    IVRSipEntities::query()->updateOrCreate(
        [
            'ivr_id' => $IVRId,
            'field' => AnnouncementTypeEnum::TIMEOUT,
        ],
        [
            'ivrable_id' => $sipEntityDto->getTimeout(),
            'ivrable_type' => $sipEntityDto->getTimeoutType(),
        ]
    );

    IVRSipEntities::query()->updateOrCreate(
        [
            'ivr_id' => $IVRId,
            'field' => AnnouncementTypeEnum::ATTEMPTS,
        ],
        [
            'ivrable_id' => $sipEntityDto->getAttempts(),
            'ivrable_type' => $sipEntityDto->getAttemptsType(),
        ]
    );

    IVRSipEntities::query()->updateOrCreate(
        [
            'ivr_id' => $IVRId,
            'field' => AnnouncementTypeEnum::SCHEDULE,
        ],
        [
            'ivrable_id' => $sipEntityDto->getTimeout(),
            'ivrable_type' => $sipEntityDto->getTimeoutType(),
```

```
            ]
        );
    }

    private function createIVRLocales(int $IVRId, IVRUpdateDTO $updateDTO): void
    {
        foreach ($updateDTO->getLocaleDTO() as $DTO) {
            /** @var IVRLocaleDto $DTO */
            IVRLocale::query()->updateOrCreate(
                [
                    'ivr_id' => $IVRId,
                    'locale' => $DTO->getLanguage()
                ],
                [
                    'main' => $DTO->isMain(),
                    'key_number' => $DTO->getKeyNumber()
                ]
            );
        }
    }

    private function updateRelatedFieldsInAsteriskForIVR(IVR $IVRFromDB,
IVRUpdateDTO $updateDTO, Domain $domain): void
    {
        $IVRSchedules = $IVRFromDB->schedules;

        if ($IVRSchedules->isNotEmpty()) {
            $IVRSchedules->each(
                function (IVRSchedule $IVRSchedule) use ($updateDTO, $domain) {
                    $ivrDto = $updateDTO->getIVRDTO();
                    $ivrAnnouncementDto = $updateDTO->getAnnouncementDTO();
                    $sipEntityDto = $updateDTO->getSipEntityDTO();

                    AsteriskIVRSchedule::query()
                        ->where('id', $IVRSchedule->asterisk_ivr_id)
                        ->update(
                            [
                                'domain_id' => $domain->getKamailioId(),
                                'ivr_number' => $ivrDto->getIVRNumber() . '.' . $IVRSchedule-
>id,
                                'timeout_announcement' => $this-
>getFileNameAnnouncementById(
                                    $ivrAnnouncementDto->getTimeoutAnnouncement()
                                ),
```

```
                    'invalid_announcement' => $this-
>getFileNameAnnouncementById(
                        $ivrAnnouncementDto->getInvalidAnnouncement()
                    ),
                    'language_announcement' => $this-
>getFileNameAnnouncementById(
                        $ivrAnnouncementDto->getLanguageAnnouncement()
                    ),
                    'attempts_announcement' => $this-
>getFileNameAnnouncementById(
                        $ivrAnnouncementDto->getAttemptsAnnouncement()
                    ),
                    'extdial_announcement' => $this-
>getFileNameAnnouncementById(
                        $ivrAnnouncementDto->getExtdialAnnouncement()
                    ),
                    'extdial_invalid_announcement' => $this-
>getFileNameAnnouncementById(
                        $ivrAnnouncementDto->getExtdialInvalidAnnouncement()
                    ),
                    'timeout_failover' => $this-
>getNumberForFailoverBySipEntityIdAndClassName(
                        $sipEntityDto->getTimeout(),
                        $sipEntityDto->getTimeoutType(),
                    ),
                    'attempts_failover' => $this-
>getNumberForFailoverBySipEntityIdAndClassName(
                        $sipEntityDto->getAttempts(),
                        $sipEntityDto->getAttemptsType(),
                    ),
                    'attempts' => $ivrDto->getAttempts(),
                    'schedule_failover' => $this-
>getNumberForFailoverBySipEntityIdAndClassName(
                        $sipEntityDto->getSchedule(),
                        $sipEntityDto->getTimeoutType(),
                    ),
                    'timeout' => $ivrDto->getTimeout(),
                    'return_code' => $ivrDto->getReturnCode(),
                    'record' => $ivrDto->isRecord(),
                    'language_default' => $this->getDefaultLanguage($updateDTO),
                    'language_change' => $ivrDto->isLanguageChange(),
                ]
            );
        }
```

```
            );
        }

        $this->updateLanguagesInAsterisk($IVRFromDB, $updateDTO, $domain);
    }

    protected function updateLanguagesInAsterisk(IVR $IVRFromDB,
IVRUpdateDTO $updateDTO, Domain $domain): void
    {
        $ivrDto = $updateDTO->getIVRDTO();
        $localeDto = $updateDTO->getLocaleDTO();

        if ($ivrDto->isLanguageChange()) {
            $IVRFromDB->fresh('locales');

            $localesInDB = [];
            foreach ($localeDto as $locale) {
                $localesInDB[] = $locale->getLanguage() . '_' . $domain->company-
>kamailio_id;
            }

            IVRLanguage::query()
                ->where('ivr_number', $ivrDto->getIVRNumber())
                ->whereNotIn('option', $localesInDB)
                ->delete();

            foreach ($IVRFromDB->locales as $locale) {
                IVRLanguage::query()
                    ->firstOrCreate(
                        [
                            'domain_id' => $domain->getKamailioId(),
                            'ivr_number' => $ivrDto->getIVRNumber(),
                            'option' => $locale->pivot->key_number,
                            'language' => $locale->slug . '_' . $domain->company->kamailio_id
                        ]
                    );
            }
        } else {
            IVRLanguage::query()
                ->where('domain_id', $domain->getKamailioId())
                ->where('ivr_number', $ivrDto->getIVRNumber())
                ->delete();
        }
    }
```

```
    /**
     * @throws Throwable
     */
    public function deleteByDomainId(int $domainId): void
    {
        $this->getByDomainId($domainId)
            ->each(function (IVR $IVR) {
                $this->delete($IVR);
            });
    }

    private function getByDomainId(int $domainId): Collection
    {
        return IVR::query()
            ->whereDomain($domainId)
            ->get();
    }

    /**
     * @throws Throwable
     */
    public function delete(IVR $ivr): bool
    {
        $this->purchasedService->deleteExtensionByExtension($ivr);

        $ivr->locales()->delete();
        $ivr->sipEntity()->delete();

        $ivr->schedules()
            ->get()
            ->each(function (IVRSchedule $schedule) {
                $data = $schedule->options()->pluck('ivr_option_id', 'id')->toArray();
                $asteriskIvrOptionIds = array_values($data);
                $ivrOptionIds = array_keys($data);

                AsteriskIVRSchedule::query()->where('id', $schedule->asterisk_ivr_id)-
>delete();
                AsteriskIVROption::query()->whereKey($asteriskIvrOptionIds)->delete();
                IVROption::query()->whereKey($ivrOptionIds)->delete();
                $schedule->announcements()->delete();
                $schedule->delete();
            });
```

```
        return $ivr->delete();
    }

    public function getConnectionsForTransaction(): array
    {
        return [
            DbConnections::default(),
            DbConnections::asteriskCfg(),
        ];
    }

    /**
     * @throws Throwable
     */
    public function cleanUpByCompany(Company $company): array
    {
        IVR::query()
            ->whereCompany($company)
            ->get()
            ->each(function (IVR $ivr) {
                $this->delete($ivr);
            });

        return [];
    }
}
```

ДОДАТОК Д.

ЛІСТИНГ МУТАЦІЇ СТВОРЕННЯ НОВОГО СЦЕНАРІЮ IVR

```
<?php

namespace App\GraphQL\Mutations\FrontOffice\IVR;

use App\Dto\IVR\IVROptionCreateOrUpdateDto;
use App\Models\Extensions\IVR;
use App\Permissions\IVR\IVROptionUpdatePermission;
use Closure;
use GraphQL\Type\Definition\ResolveInfo;
use Rebing\GraphQL\Support\SelectFields;
use Throwable;

class IVROptionUpdateMutation extends IVROptionBaseMutation
{
   public const NAME = 'IVROptionUpdateMutation';

   public function authorize($root, array $args, $ctx, ResolveInfo $info = null, Closure
$fields = null): bool
   {
      return $this->can(IVROptionUpdatePermission::KEY);
   }

   /**
    * @throws Throwable
    */
   public function doResolve(
      $root,
      array $args,
      $context,
      ResolveInfo $info,
      SelectFields $fields
   ): IVR {
      return makeTransaction(
         fn() => $this->service-
>updateOrCreate(IVROptionCreateOrUpdateDto::byArgs($args)),
         $this->service->getConnectionsForTransaction()
      );
   }
}
```

ДОДАТОК Е.

ЛІСТИНГ СЕРВІСУ ДЛЯ РОБОТИ ЗІ СЦЕНАРІЄМ IVR

```
<?php

namespace App\Services\IVR;

use App\Dto\IVR\IVROptionCreateOrUpdateDto;
use App\Dto\IVR\IVROptionDto;
use App\Dto\IVR\IVRScheduleDto;
use App\Enums\Audio\AnnouncementTypeEnum;
use App\Enums\IVR\IVRFailoverConfigEnum;
use App\Enums\IVR\IVRFailoverTypesEnum;
use App\Helpers\DbConnections;
use App\Models\AsteriskCfg\IVR as AsteriskIVR;
use App\Models\AsteriskCfg\IVROption as AsteriskIVROption;
use App\Models\Extensions\IVR;
use App\Models\Extensions\IVRAnnouncement;
use App\Models\Extensions\IVROption;
use App\Models\Extensions\IVRSchedule;
use Illuminate\Support\Facades\Log;
use Throwable;

class IVROptionService extends BaseIVRService
{
    /**
     * @throws Throwable
     */
    public function updateOrCreate(IVROptionCreateOrUpdateDto $dto): IVR
    {
        $ivr = IVR::query()
            ->with(['domain', 'schedules', 'announcements', 'sipEntity'])
            ->findOrFail($dto->getIVRId());

        $existsModelsIds = [];

        foreach ($dto->getIVRScheduleDto() as $scheduleDto) {
            $existsModelsIds[] = $scheduleDto->getId();
        }

        $this->deleteMissedIVRSchedulesAndRelatedData($ivr, $existsModelsIds);

        foreach ($dto->getIVRScheduleDto() as $scheduleDto) {
            $IVRSchedule = IVRSchedule::query()->updateOrCreate(
```

```
            [
                'id' => $scheduleDto->getId(),
                'ivr_id' => $ivr->id,
            ],
            $scheduleDto->getFullSchedule()
        );

        $IVRNumber = $ivr->ivr_number . '.' . $IVRSchedule->id;

        $this->updateOrCreateIVRAnnouncements($IVRSchedule->id, $scheduleDto);

        $asteriskIVR = AsteriskIVR::updateOrCreate(
            [
                'id' => $IVRSchedule->asterisk_ivr_id
            ],
            array_merge(
                $scheduleDto->getScheduleWithoutType(),
                [
                    'domain_id' => $ivr->domain->getKamailioId(),
                    'ivr_number' => $IVRNumber,
                    'announcement' => $this->getFileNameAnnouncementById(
                        $IVRSchedule->announcement->id
                    ),
                    'timeout_announcement' => $this->getFileNameAnnouncementById(
                        $ivr->timeout_announcement->id
                    ),
                    'invalid_announcement' => $this->getFileNameAnnouncementById(
                        $ivr->invalid_announcement->id
                    ),
                    'language_announcement' => $this->getFileNameAnnouncementById(
                        $ivr->language_announcement->id
                    ),
                    'attempts_announcement' => $this->getFileNameAnnouncementById(
                        $ivr->attempts_announcement->id
                    ),
                    'extdial_announcement' => $this->getFileNameAnnouncementById(
                        $ivr->extdial_announcement->id
                    ),
                    'extdial_invalid_announcement' => $this-
>getFileNameAnnouncementById(
                        $ivr->extdial_invalid_announcement->id
                    ),
                    'timeout_failover' => $this-
>getNumberForFailoverBySipEntityIdAndClassName(
```

```
                    $ivr->timeout_failover_id,
                    IVRFailoverConfigEnum::CONFIG[$ivr->timeout_failover_type] ??
null,
                ),
                'attempts_failover' => $this-
>getNumberForFailoverBySipEntityIdAndClassName(
                    $ivr->attempt_failover_id,
                    IVRFailoverConfigEnum::CONFIG[$ivr->attempt_failover_type] ??
null,
                ),
                'attempts' => $ivr->attempts,
                'schedule_failover' => $this-
>getNumberForFailoverBySipEntityIdAndClassName(
                    $ivr->schedule_failover_id,
                    IVRFailoverConfigEnum::CONFIG[$ivr->schedule_failover_type]
?? null,
                ),
                'timeout' => $ivr->timeout,
                'return_code' => $ivr->return_code,
                'record' => $ivr->record,
                'language_default' => $ivr->locales->where('main', true)->first()->slug,
                'language_change' => $ivr->language_change,
            ]
        )
    );

    $IVRSchedule->update(
        [
            'asterisk_ivr_id' => $asteriskIVR->id
        ]
    );

    $existsModelsUUIDs = [];
    foreach ($scheduleDto->getOptions() as $optionDto) {
        $existsModelsUUIDs[] = $optionDto->getUuid();
    }
    $modelsFromDBToDelete = IVROption::select(['id', 'ivr_option_id'])
        ->where('ivr_schedule_id', $IVRSchedule->id)
        ->whereNotIn('uuid', $existsModelsUUIDs)
        ->get();

    IVROption::whereIn('id', $modelsFromDBToDelete->pluck('id')->toArray())
        ->delete();
```

```
        AsteriskIVROption::whereIn('id', $modelsFromDBToDelete-
>pluck('ivr_option_id')->toArray())
            ->delete();

        /** @var IVROptionDto $optionDto */
        foreach ($scheduleDto->getOptions() as $optionDto) {
            $IVROption = IVROption::query()->where('uuid', $optionDto->getUuid())-
>first();

            $asteriskIVROption = AsteriskIVROption::updateOrCreate(
                [
                    'id' => optional($IVROption)->ivr_option_id
                ],
                array_merge(
                    $scheduleDto->getScheduleWithoutType(),
                    [
                        'domain_id' => $ivr->domain->getKamailioId(),
                        'ivr_number' => $IVRNumber,
                        'option' => $optionDto->getOption(),
                        'destination' => $this->getNumberBySipEntityIdAndClassName(
                            $optionDto->getDestinationId(),
                            $optionDto->getDestinationType()
                        ),
                        'announcement' => $this-
>getFileNameAnnouncementById($optionDto->getAnnouncement()),
                        'failover' => $this->getNumberBySipEntityIdAndClassName(
                            $optionDto->getFailoverId(),
                            $optionDto->getFailoverType()
                        ),
                        'extdial' => $optionDto->getExtdial(),
                        'lang' => $optionDto->getLang(),
                    ]
                )
            );

            IVROption::updateOrCreate(
                [
                    'ivr_schedule_id' => $IVRSchedule->id,
                    'uuid' => $optionDto->getUuid(),
                ],
                array_merge(
                    $scheduleDto->getScheduleWithoutType(),
                    [
                        'parent_uuid' => $optionDto->getParentUUID(),
```

```
                    'ivr_option_id' => $asteriskIVROption->id,
                    'announcement_id' => $optionDto->getAnnouncement(),
                    'position' => $optionDto->getPosition(),
                    'option' => $optionDto->getOption(),
                    'destination_id' => $optionDto->getDestinationId(),
                    'destination_type' => $optionDto->getDestinationType() ??
IVRFailoverTypesEnum::END_CALL,
                    'failoverable_id' => $optionDto->getFailoverId(),
                    'failoverable_type' => $optionDto->getFailoverType() ??
IVRFailoverTypesEnum::END_CALL,
                    'schedule_type' => $scheduleDto->getType(),
                    'extdial' => $optionDto->getExtdial(),
                    'lang' => $optionDto->getLang(),
                ]
            )
        );
    }
}

return $ivr->fresh(['domain', 'schedules', 'announcements', 'sipEntity']);
}

public function deleteMissedIVRSchedulesAndRelatedData(IVR $IVR, array
$IVRScheduleIds): void
{
    $modelsFromDBToDelete = IVRSchedule::query()
        ->select(['id', 'asterisk_ivr_id'])
        ->where('ivr_id', $IVR->id)
        ->whereNotIn('id', $IVRScheduleIds)
        ->get();

    $asteriskIVROptionIds = IVROption::query()
        ->select('ivr_option_id')
        ->whereIn('ivr_schedule_id', $modelsFromDBToDelete->pluck('id')-
>toArray())
        ->get()
        ->pluck('ivr_option_id')
        ->toArray();

    IVROption::query()
        ->whereIn('ivr_schedule_id', $modelsFromDBToDelete->pluck('id')-
>toArray())
        ->delete();
```

```
        IVRAnnouncement::query()
            ->whereIn('announced_id', $modelsFromDBToDelete->pluck('id')->toArray())
            ->where('announced_type', IVRSchedule::MORPH_NAME)
            ->delete();

        AsteriskIVR::query()
            ->whereIn('id', $modelsFromDBToDelete->pluck('asterisk_ivr_id')->toArray())
            ->delete();

        AsteriskIVROption::query()
            ->whereIn('id', $asteriskIVROptionIds)
            ->delete();

        IVRSchedule::query()
            ->whereNotIn('id', $IVRScheduleIds)
            ->delete();
    }

    private function updateOrCreateIVRAnnouncements(int $IVRScheduleId,
IVRScheduleDto $dto): void
    {
        IVRAnnouncement::updateOrCreate(
            [
                'announced_id' => $IVRScheduleId,
                'announced_type' => IVRSchedule::MORPH_NAME,
                'field' => AnnouncementTypeEnum::ANNOUNCEMENT,
            ],
            [
                'announcement_id' => $dto->getAnnouncement(),
            ]
        );
    }

    private function getNumberBySipEntityIdAndClassName(?int $id, ?string
$className): ?string
    {
        if (is_null($id) || is_null($className)) {
            return null;
        }

        $instance = new $className();

        try {
            return $instance::select('number')->findOrFail($id)->getAttribute('number');
```

```
        } catch (Throwable $exception) {
            Log::error(
                "Error while fetching data from " . $instance->getTable() . " table by ID = " .
$id,
                [
                    'message' => $exception->getMessage()
                ]
            );
        }

        return null;
    }

    public function getConnectionsForTransaction(): array
    {
        return [
            DbConnections::default(),
            DbConnections::asteriskCfg(),
        ];
    }
}
```

ДОДАТОК :.

ЛІСТИНГ КОДУ АВТЕНТИФІКАЦІЇ

```
<?php

namespace App\GraphQL\Mutations;

use App\Enums\Messages\AuthorizationMessageEnum;
use App\Rules\LoginAdmin;
use Closure;
use GraphQL\Type\Definition\ResolveInfo;
use GraphQL\Type\Definition\Type;
use Rebing\GraphQL\Support\SelectFields;

abstract class BaseLoginMutation extends BaseMutation
{
    public function authorize(
        mixed $root,
        array $args,
        mixed $ctx,
        ResolveInfo $info = null,
        Closure $fields = null
    ): bool {
        return $this->getAuthGuard()->guest();
    }

    public function getAuthorizationMessage(): string
    {
        return AuthorizationMessageEnum::AUTHORIZED;
    }

    public function args(): array
    {
        return [
            'username' => Type::nonNull(Type::string()),
            'password' => Type::nonNull(Type::string()),
        ];
    }

    public function doResolve(
        mixed $root,
        array $args,
        mixed $context,
        ResolveInfo $info,
```

```
        SelectFields $fields
    ): array {
        return $this->passportService->auth($args['username'], $args['password']);
    }

    protected function rules(array $args = []): array
    {
        return [
            'username' => ['required', 'email'],
            'password' => ['required', 'string', 'min:8', new LoginAdmin($args)],
        ];
    }

}
```

ДОДАТОК К.

ЛІСТИНГ МОДУЛЯ ТЕСТУВАННЯ

```
<?php

namespace Tests\Feature\Mutations\FrontOffice\IVR;

use App\Enums\IVR\IVRFailoverConfigEnum;
use App\Enums\IVR\IVRFailoverTypesEnum;
use App\GraphQL\Mutations\FrontOffice\IVR\IVRUpdateMutation;
use App\GraphQL\Types\AudioFiles\CommonAnnouncementType;
use App\GraphQL\Types\AudioFiles\CompanyAnnouncementType;
use App\Helpers\DbConnections;
use App\Models\AsteriskCfg\IVRLanguage;
use App\Models\Audio\Announcement;
use App\Models\Audio\AnnouncementTranslates;
use App\Models\Companies\Company;
use App\Models\Extensions\Domain;
use App\Models\Extensions\IVR;
use App\Models\Extensions\IVRAnnouncement;
use App\Models\Extensions\IVRLocale;
use App\Models\Extensions\IVRSipEntities;
use App\Models\Extensions\SipGroup;
use App\Models\Extensions\SipUser;
use App\Models\Localization\Locale;
use App\Models\Users\User;
use App\Permissions\IVR\IVRUpdatePermission;
use Database\Factories\IVR\GraphQLBuilder;
use Database\Factories\IVR\GraphQLEnumValue;
use Illuminate\Database\Eloquent\Relations\Relation;
use Illuminate\Foundation\Testing\DatabaseTransactions;
use Illuminate\Support\Str;
use Illuminate\Testing\TestResponse;
use Tests\TestCase;
use Tests\Traits\Permissions\RoleHelper;

class IVRUpdateTest extends TestCase
{
    use DatabaseTransactions;
    use RoleHelper;

    public const MUTATION = IVRUpdateMutation::NAME;

    protected User $firstUser;
```

```
protected User $secondUser;

protected Domain $firstDomain;
protected Domain $secondDomain;
protected array $data = [];
private SipUser $firstSipUser;
private SipUser $secondSipUser;
private SipGroup $firstSipGroup;
private SipGroup $secondSipGroup;
private Announcement $firstAnnouncement;
private Announcement $secondAnnouncement;
private IVR $IVR;

public function test_cant_update_ivr_by_guest(): void
{
   $result = $this->query()
      ->assertOk();

   $this->assertGraphQlUnauthorized($result);
}

private function query(): TestResponse
{
   $query = $this->makeBuilderFromData()->getQuery();

   return $this->postGraphQL(['query' => $query]);
}

private function makeBuilderFromData(): GraphQLBuilder
{
   return GraphQLBuilder::mutation(static::MUTATION)
      ->select(
         'id',
         'name',
         'description',
         'ivr_number',
         'domain_id',
         'attempts',
         'timeout',
         'return_code',
         'record',
         'timeout_failover_id',
         'timeout_failover_type',
         'language_change',
```

```
            'language_default',
        )
        ->unionOn('timeout_announcement', [
            CommonAnnouncementType::NAME => ['name'],
            CompanyAnnouncementType::NAME => ['name'],
        ])
        ->unionOn('invalid_announcement', [
            CommonAnnouncementType::NAME => ['name'],
            CompanyAnnouncementType::NAME => ['name'],
        ])
        ->unionOn('language_announcement', [
            CommonAnnouncementType::NAME => ['name'],
            CompanyAnnouncementType::NAME => ['name'],
        ])
        ->unionOn('attempts_announcement', [
            CommonAnnouncementType::NAME => ['name'],
            CompanyAnnouncementType::NAME => ['name'],
        ])
        ->unionOn('extdial_announcement', [
            CommonAnnouncementType::NAME => ['name'],
            CompanyAnnouncementType::NAME => ['name'],
        ])
        ->unionOn('extdial_invalid_announcement', [
            CommonAnnouncementType::NAME => ['name'],
            CompanyAnnouncementType::NAME => ['name'],
        ])
        ->select(['locales' => ['main', 'slug', 'name']])
        ->where('id', $this->data['id'])
        ->where('domain_id', $this->data['domain_id'])
        ->where('name', $this->data['name'])
        ->where('description', $this->data['description'])
        ->where('ivr_number', $this->data['ivr_number'])
        ->where('timeout_failover', $this->data['timeout_failover'])
        ->where('timeout_failover_type', $this->data['timeout_failover_type'])
        ->where('timeout_announcement', $this->data['timeout_announcement'])
        ->where('invalid_announcement', $this->data['invalid_announcement'])
        ->where('language_announcement', $this->data['language_announcement'])
        ->where('attempts_announcement', $this->data['attempts_announcement'])
        ->where('extdial_announcement', $this->data['extdial_announcement'])
        ->where('extdial_invalid_announcement', $this-
>data['extdial_invalid_announcement'])
        ->where('attempts', $this->data['attempts'])
        ->where('timeout', $this->data['timeout'])
        ->where('record', $this->data['record'])
```

```
        ->where('languages', $this->data['languages'])
        ->where('return_code', $this->data['return_code'])
        ->where('language_change', $this->data['language_change']);
}

public function test_cant_update_ivr_by_user_without_permission(): void
{
    $this->loginAsUser($this->firstUser);

    $result = $this->query()
        ->assertOk();

    $this->assertGraphQlUnauthorized($result);
}

public function test_can_update_ivr_by_user_with_permission(): void
{
    $this->firstUser
        ->assignRole(
            $this->generateRole(
                'Locale IVR user',
                [
                    IVRUpdatePermission::KEY,
                ]
            )
        );

    $this->loginAsUser($this->firstUser);

    $this->complexAssertAnnouncementNotAttached();
    $this->assertCommonSettingsNotAttached();
    $this->complexAssertLocalesNotAttached();
    $this->complexAssertFailoversNotAttached();

    $this->updateAndCheck();

    $this->data['language_change'] = true;
    $this->data['languages'][0]['key_number'] = '2';
    $this->data['languages'][] = [
        'is_main' => false,
        'locale' => 'en',
        'key_number' => '1'
    ];
```

```
        $this->updateAndCheck();

        $this->data['languages'][0]['key_number'] = null;
        unset($this->data['languages'][1]);
        $this->data['language_change'] = false;

        $this->updateAndCheck();

        $this->data['name'] = 'updated name';
        $this->data['description'] = 'updated description';
        $this->data['ivr_number'] = 'ivr_number';
        $this->data['attempts'] = 5;
        $this->data['return_code'] = '9';
        $this->data['record'] = false;

        $this->updateAndCheck();

        $this->data['timeout_failover'] = $this->firstSipGroup->id;
        $this->data['timeout_failover_type'] =
GraphQLEnumValue::VALUE(IVRFailoverTypesEnum::GROUP);

        $this->updateAndCheck();

        $this->data['timeout_failover_type'] =
GraphQLEnumValue::VALUE(IVRFailoverTypesEnum::END_CALL);
        $this->data['timeout_failover'] = null;

        $this->updateAndCheck();
    }

    public function complexAssertAnnouncementNotAttached(): void
    {
        $this->assertAnnouncementNotAttached($this->data['timeout_announcement'],
'timeout');
        $this->assertAnnouncementNotAttached($this->data['invalid_announcement'],
'invalid');
        $this->assertAnnouncementNotAttached($this->data['language_announcement'],
'language');
        $this->assertAnnouncementNotAttached($this->data['attempts_announcement'],
'attempts');
        $this->assertAnnouncementNotAttached($this->data['extdial_announcement'],
'extdial');
        $this->assertAnnouncementNotAttached($this-
>data['extdial_invalid_announcement'], 'extdial_invalid');
```

```
}

public function assertAnnouncementNotAttached(int $id, string $type = null): void
{
    $this->assertDatabaseHas(Announcement::TABLE, [
        'id' => $id
    ]);

    $this->assertDatabaseMissing(IVRAnnouncement::TABLE, [
        'announced_id' => $this->data['id'],
        'announced_type' => IVR::MORPH_NAME,
        'announcement_id' => $id,
        'field' => $type
    ]);
}

public function assertCommonSettingsNotAttached(): void
{
    $this->assertDatabaseMissing(IVR::TABLE, [
        'id' => $this->data['id'],
        'name' => $this->data['name'],
        'description' => $this->data['description'],
        'ivr_number' => $this->data['ivr_number'],
        'domain_id' => $this->data['domain_id'],
        'attempts' => $this->data['attempts'],
        'timeout' => $this->data['timeout'],
        'return_code' => $this->data['return_code'],
        'record' => $this->data['record'],
        'language_change' => $this->data['language_change'],
    ]);
}

public function complexAssertLocalesNotAttached(): void
{
    foreach ($this->data['languages'] as $language) {
        $this->assertDatabaseHas(Locale::TABLE, [
            'slug' => $language['locale']
        ]);

        $this->assertDatabaseMissing(IVRLocale::TABLE, [
            'locale' => $language['locale'],
            'ivr_id' => $this->IVR->id,
            'main' => $language['is_main'],
            'key_number' => $language['key_number'],
```

```
            ]);

            $this->assertDatabaseMissing(IVRLanguage::TABLE, [
                'ivr_number' => $this->data['ivr_number'],
                'option' => $language['key_number'],
                'language' => $language['locale'] . '_' . $this->firstUser->company-
>kamailio_id,
            ],                      DbConnections::ASTERISK_CFG);
        }
    }

    public function complexAssertFailoversNotAttached(): void
    {
        $timeoutFailoverType = $this->data['timeout_failover_type']->value ??
IVRFailoverTypesEnum::END_CALL;
        $tableNameForCheckExistsFailover =
IVRFailoverConfigEnum::CONFIG_FOR_VALIDATION_TABLE[$timeoutFailover
Type] ?? null;

        if ($tableNameForCheckExistsFailover) {
            $this->assertDatabaseHas($tableNameForCheckExistsFailover, [
                'id' => $this->data['timeout_failover']
            ]);
        }
        if ($timeoutFailoverType !== IVRFailoverTypesEnum::END_CALL) {
            $this->assertDatabaseMissing(IVRSipEntities::TABLE, [
                'ivr_id' => $this->data['id'],
                'ivrable_id' => $this->data['timeout_failover'],
                'ivrable_type' => Relation::getMorphedModel($timeoutFailoverType),
                'field' => 'timeout',
            ]);
        } else {
            $this->assertDatabaseMissing(IVRSipEntities::TABLE, [
                'ivr_id' => $this->data['id'],
                'ivrable_id' => $this->data['timeout_failover'],
                'ivrable_type' => null,
                'field' => 'timeout',
            ]);
        }
    }

    public function updateAndCheck(bool $needDump = false): void
    {
        $needDump
```

```
            ? $this->query()->dump()->assertOk()
            : $this->query()->assertOk();

        $this->complexAssertAnnouncementAttached();
        $this->assertCommonSettingsAttached();
        $this->complexAssertLocalesAttached();
        $this->complexAssertFailoversAttached();
    }

    public function complexAssertAnnouncementAttached(): void
    {
        $this->assertAnnouncementAttached($this->data['timeout_announcement'],
'timeout');
        $this->assertAnnouncementAttached($this->data['invalid_announcement'],
'invalid');
        $this->assertAnnouncementAttached($this->data['language_announcement'],
'language');
        $this->assertAnnouncementAttached($this->data['attempts_announcement'],
'attempts');
        $this->assertAnnouncementAttached($this->data['extdial_announcement'],
'extdial');
        $this->assertAnnouncementAttached($this-
>data['extdial_invalid_announcement'], 'extdial_invalid');
    }

    public function assertAnnouncementAttached(int $id, string $type = null): void
    {
        $this->assertDatabaseHas(Announcement::TABLE, [
            'id' => $id
        ]);

        $this->assertDatabaseHas(IVRAnnouncement::TABLE, [
            'announced_id' => $this->data['id'],
            'announced_type' => IVR::MORPH_NAME,
            'announcement_id' => $id,
            'field' => $type
        ]);
    }

    public function assertCommonSettingsAttached(): void
    {
        $this->assertDatabaseHas(IVR::TABLE, [
            'id' => $this->data['id'],
            'name' => $this->data['name'],
```

```
            'description' => $this->data['description'],
            'ivr_number' => $this->data['ivr_number'],
            'domain_id' => $this->data['domain_id'],
            'attempts' => $this->data['attempts'],
            'timeout' => $this->data['timeout'],
            'return_code' => $this->data['return_code'],
            'record' => $this->data['record'] ? 1 : 0,
            'language_change' => $this->data['language_change'] ? 1 : 0,
        ]);
    }

    public function complexAssertLocalesAttached(): void
    {
        if ($this->data['language_change']) {
            $this->assertDatabaseCount(
                IVRLanguage::TABLE,
                count($this->data['languages']),
                DbConnections::ASTERISK_CFG
            );
        } else {
            $this->assertDatabaseCount(
                IVRLanguage::TABLE,
                0,
                DbConnections::ASTERISK_CFG
            );
        }

        foreach ($this->data['languages'] as $language) {
            $this->assertDatabaseHas(Locale::TABLE, [
                'slug' => $language['locale']
            ]);

            $this->assertDatabaseHas(IVRLocale::TABLE, [
                'locale' => $language['locale'],
                'ivr_id' => $this->IVR->id,
                'main' => $language['is_main'],
                'key_number' => $language['key_number'],
            ]);

            if ($this->data['language_change']) {
                $this->assertDatabaseHas(
                    IVRLanguage::TABLE,
                    [
                        'ivr_number' => $this->data['ivr_number'],
```

```
                    'option' => $language['key_number'],
                    'language' => $language['locale'] . '_' . $this->firstUser->company->kamailio_id,
                ],
                DbConnections::ASTERISK_CFG
            );
        }
    }
}

public function complexAssertFailoversAttached(): void
{
    $timeoutFailoverType = $this->data['timeout_failover_type']->value ?? IVRFailoverTypesEnum::END_CALL;
    $tableNameForCheckExistsFailover = IVRFailoverConfigEnum::CONFIG_FOR_VALIDATION_TABLE[$timeoutFailoverType] ?? null;

    if ($tableNameForCheckExistsFailover) {
        $this->assertDatabaseHas($tableNameForCheckExistsFailover, [
            'id' => $this->data['timeout_failover']
        ]);
    }
    if ($timeoutFailoverType !== IVRFailoverTypesEnum::END_CALL) {
        $this->assertDatabaseHas(IVRSipEntities::TABLE, [
            'ivr_id' => $this->data['id'],
            'ivrable_id' => $this->data['timeout_failover'],
            'ivrable_type' => Relation::getMorphedModel($timeoutFailoverType),
            'field' => 'timeout',
        ]);
    } else {
        $this->assertDatabaseHas(IVRSipEntities::TABLE, [
            'ivr_id' => $this->data['id'],
            'ivrable_id' => $this->data['timeout_failover'],
            'ivrable_type' => null,
            'field' => 'timeout',
        ]);
    }
}

public function setUp(): void
{
    parent::setUp();
```

```
        $this->fillRelatedData();
    }

    private function fillRelatedData(): void
    {
        $firstCompany = Company::factory()->create();
        $secondCompany = Company::factory()->create();

        $this->firstUser = User::factory()->withCompany($firstCompany)->create();
        $this->secondUser = User::factory()->withCompany($secondCompany)-
>create();

        $this->firstDomain = Domain::factory()->withCompany($firstCompany)-
>create();
        $this->secondDomain = Domain::factory()->withCompany($firstCompany)-
>create();

        $this->firstSipUser = SipUser::factory()->create(
            [
                'domain_id' => $this->firstDomain->id,
                'user_id' => $this->firstUser->id
            ]
        );

        $this->secondSipUser = SipUser::factory()->create(
            [
                'domain_id' => $this->secondDomain->id,
                'user_id' => $this->secondUser->id
            ]
        );

        $this->firstSipGroup = SipGroup::factory()->create(
            [
                'domain_id' => $this->firstDomain->id,
            ]
        );

        $this->secondSipGroup = SipGroup::factory()->create(
            [
                'domain_id' => $this->secondDomain->id,
            ]
        );

        $this->firstAnnouncement = Announcement::factory()->create(
```

```
            [
                'company_id' => $firstCompany->id
            ]
        );

        AnnouncementTranslates::create(
            [
                'row_id' => $this->firstAnnouncement->id,
                'locale' => 'ru',
                'path' => 'test_path',
            ]
        );

        AnnouncementTranslates::create(
            [
                'row_id' => $this->firstAnnouncement->id,
                'locale' => 'en',
                'path' => 'test_path',
            ]
        );

        $this->secondAnnouncement = Announcement::factory()->create(
            [
                'company_id' => $secondCompany->id
            ]
        );

        $this->IVR = IVR::factory()->create(['domain_id' => $this->firstDomain->id]);

        $this->data['id'] = $this->IVR->id;
        $this->data['domain_id'] = $this->secondDomain->id;
        $this->data['name'] = 'Test ivr name';
        $this->data['description'] = 'Test ivr description';
        $this->data['ivr_number'] = Str::random(5);
        $this->data['timeout_failover'] = $this->firstSipUser->id;
        $this->data['timeout_failover_type'] =
GraphQLEnumValue::VALUE(IVRFailoverTypesEnum::EMPLOYEE);
        $this->data['timeout_announcement'] = $this->firstAnnouncement->id;
        $this->data['invalid_announcement'] = $this->firstAnnouncement->id;
        $this->data['language_announcement'] = $this->firstAnnouncement->id;
        $this->data['attempts_announcement'] = $this->firstAnnouncement->id;
        $this->data['extdial_announcement'] = $this->firstAnnouncement->id;
        $this->data['extdial_invalid_announcement'] = $this->firstAnnouncement->id;
        $this->data['attempts'] = 3;
```

```
        $this->data['timeout'] = 10;
        $this->data['record'] = true;
        $this->data['languages'] = [
            [
                'is_main' => true,
                'locale' => 'ru',
                'key_number' => null
            ]
        ];
        $this->data['return_code'] = '*';
        $this->data['language_change'] = false;
    }
}
```