Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо–кваліфікаційний рівень)

на тему: Розробка методів реактивного виведення даних для веб-сервісу зберігання та відтворення аудіофайлів

Виконав: студент II курсу, групи 1ПІ–18м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)

Корягін І. С.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. КН Іванчук Я.В.
(прізвище та ініціали)

ВНТУ– 2019

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Освітньо–кваліфікаційний рівень – магістр
Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
“____”_____2019 року

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

Корягіну Іллі Сергійовичу

1. Тема роботи: Розробка методів реактивного виведення даних для веб-сервісу зберігання та відтворення аудіофайлів
керівник роботи: к.т.н., доцент кафедри ПЗ Ракитянська Г.Б. затверджені наказом вищого навчального закладу від “___”_____20__ року №__

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи : базові методи виведення даних, метод локального збереження даних, веб-сервіс зберігання та відтворення аудіофайлів, фреймворк Angular, мова програмування TypeScript.

4.Зміст розрахунково–пояснювальної записки (перелік питань, які потрібно розробити): вступ; аналіз та постановка задачі; розробка архітектури методів реактивного виведення даних; розробка методів реактивного виведення даних; тестування методів, висновки; список використаних джерел, додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):
демонстрація роботи аналогів, діаграми послідовності, класів, структурна карта Константайна, структурна схема компонентів додатку, спроектоване сховище даних, реактивне виведення даних користувачу, тестування додатку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–4	к.т.н. Ракитянська Г. Б., доцент кафедри ПЗ		
5	Бальзан М.В., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз проблеми, обґрунтування актуальності розробки методів та постановка задачі	07.10.2019 – 27.10.2019	Виконано
2	Розробка архітектури методів реактивного виведення даних	28.10.2019 – 8.11.2019	Виконано
3	Розробка методів реактивного виведення даних	9.11.2019 – 20.11.2019	Виконано
4	Тестування методів	21.11.2019 – 3.12.2019	Виконано
5	Економічна частина	4.12.2019 – 10.12.2019	Виконано

Студент _____ **Корягін І.С.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Ракитянська Г.Б.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістирській кваліфікаційній роботі розглянуто методи реактивного виведення даних. Отримав подальший розвиток метод селективного вибору даних, який ґрунтується на потоковому виведенні даних з проміжного сховища даних, а також забезпечує автоматичне відслідковування змін.

З використанням середовища програмування WebStorm мовою програмування TypeScript з використанням фреймворку Angular розроблено методи реактивного виведення даних для веб-сервісу зберігання і відтворення аудіофайлів.

ANNOTATION

The master's qualification work deals with methods of reactive data output. The method of selective data selection, which is based on the streaming of data from an intermediate data warehouse, and provides automatic tracking of changes, has been further developed.

Using the WebStorm programming environment in the TypeScript programming language using the Angular framework, methods for reactive data output for a web service for storing and playing audio files were developed.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. В сучасному світі важко уявити життя людини без музики. З раннього дитинства нас оточують різні звуки такі як материнський голос, спів пташок чи шелест листя, ми чуємо різні мелодії, колискові, народні пісні і майже з самого народження нас оточує музика. Всім відомо, що музика повністю впливає на стан та поведінку людини, робить її спокійнішою чи роздратованішою, щасливішою чи нещасною, радісною чи засмученою, оскільки як відомо людина складається на 80% з води, на яку музика має безпосередній вплив.

Можливість передавати думки та емоції людини засобами музичного мистецтва обґрунтовується фізично та біологічно зумовленим зв'язком звукових проявів людини з її психічною діяльністю (особливо емоційною), а також активності звуку як подразника та сигналу до дії. В деякому сенсі музика вважається аналогічною до мовлення людини, коли її внутрішній стан та ставлення до чогось виражається через зміну характеристик звучання голосу при висловлюванні.

На сьогоднішній день багато батьків пов`язують майбутнє своїх дітей з музикою. Є корисним, якщо дитина змалечку починає вивчати гру на музичних інструментах, вивчаючи музичні твори великих і відмових композиторів таких як Бах, Бетховен, Моцарт, Паганіні. Ранній музичний розвиток дозволяє швидко знаходити захоплення та натхнення, бути більш творчим, працювати над собою і загартовувати свій характер. Проте, також існує багато людей, які просто люблять музику і дня прожити без неї не можуть, не маючи музичної освіти.

Потреба у поширенні музичних творів зумовила розробку великої кількості веб-сервісів для прослуховування, купівлі і скачування аудіофайлів.

Проте значною проблемою більшості веб-сервісів є перевантаженість і низька швидкодія через низький рівень оптимізації сервісу, одним із проявів якого є надмірна кількість запитів до серверу. Щоб уникнути таких проблем в процесі розробки веб-сервісу необхідно розробити методи реактивного виведення даних, які б дозволяли централізовано відслідковувати зміну

отриманих даних з серверу і реактивно, тобто без зайвих нових запитів до серверу, відображати їх клієнту.

Також актуальним залишається питання швидкодії, оскільки веб-сервіс для зберігання та відтворення аудіофайлів працює з великими об`ємами даних через що відповідь з серверу може займати достатньо багато часу, тому враховуючи це необхідно забезпечити мінімально можливу кількість запитів до серверу, яка б була максимально ефективною для усіх компонентів, в яких використовуються отриманні з серверу дані.

Задача мінімізації часу очікування даних користувачем потребує вирішення в рамках цього дослідження, оскільки сучасний ритм життя та розвиток технологій не передбачає надмірного очікування завантаження і виведення даних клієнту.

Всі вищеперераховані недоліки можливо усунути завдяки розробці методів реактивного виведення даних для веб-сервісу зберігання та відтворення аудіофайлів.

Таким чином, розробка методів реактивного виведення даних для веб-сервісу зберігання та відтворення аудіофайлів є актуальною на сьогоднішній день.

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

Мета та завдання дослідження. Метою роботи є підвищення продуктивності веб-сервісу зберігання та відтворення аудіофайлів за рахунок реактивного виведення даних.

Основними задачами дослідження є:

- провести аналіз існуючих методів і засобів реактивного виведення даних;
- розробити методи реактивного виведення даних для музичного веб-сервісу «myMusic»;
- розробити програмні компоненти веб-сервісу зберігання та відтворення аудіо файлів з реактивним виведенням;

- провести експериментальні дослідження ефективності розроблених методів.

Об’єкт дослідження – процес реактивного виведення даних.

Предмет дослідження – методи і засоби реактивного виведення даних.

Методи дослідження. У процесі досліджень використовувались: теорія алгоритмів та структур даних, методи інтернет-технологій та теорія баз даних, веб-програмування для розробки методів реактивного виведення та селекції даних; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Вперше запропоновано метод реактивного виведення даних, особливість якого полягає у збереженні даних в проміжному сховищі у вигляді структурованого об’єкту, що забезпечує зменшення кількості запитів до серверу.
2. Подальшого розвитку отримав метод селективного вибору отриманих даних, у якому, на відміну існуючих, відслідковуються зміни даних проміжного сховища, що забезпечує автоматичне оновлення вибраних даних.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі результатів запропоновано алгоритми та розроблено програмні засоби веб-сервісу зберігання та відтворення аудіофайлів.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто.

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’ятьох розділів, висновків, списку використаних джерел, що містить 22 найменування, 4 додатків. Робота містить 32 ілюстрації, 8 таблиці. У першому розділі роботи проаналізовано предметну область, розглянуто методи виведення даних та проблеми оптимізації веб-сервісу. В результаті, виявлено доцільність розробки методів реактивного виведення даних для веб-сервісу «myMusic». У другому розділі розроблено загальну структуру методу реактивного виведення даних. У третьому розділі в результаті аналізу мов

програмування та середовищ розробки обрано, відповідно, TypeScript та WebStorm, оскільки вони надають необхідні можливості для реалізації методу реативного виведення даних. У четвертому розділі розглянуто підходи до тестування програми та проведено тестування веб-сервісу, протестовано основні можливості розроблених модулів. У п’ятому розділі проведено економічне обґрунтування доцільності та економічної вигоди даного проекту. У додатках міститься технічне завдання на роботу, лістинг коду та ілюстративний матеріал до захисту роботи.

1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз стану проблеми

Суспільство тісно пов’язано з музикою, яка по-різному діє на кожну людину: вона змінює наш настрій, дає нам натхнення, вона розважає і заспокоює нас, нагадує нам про різні моменти нашого минулого. Музика безперервно змінюється, її не можливо зупинити, можливо вплинути лише на її розвиток, тому вона розвивається з виходом кожної нової пісні. Музика – це насамперед творчість і індивідуальне самовираження кожної людини – внаслідок цього сформувалось безліч жанрів і піджанрів музики, до яких щороку додається декілька нових видів. Відомо, що сприйняття музики різнобічне: кожна людина має свій особистий відбиток, що залишається після прослуховануванння будь-якої пісні. Саме тому, люди і полюбляють музику, бо знаходять у ній щось своє.

Музика будує особливу субкультуру, змінює погляди людей на багато речей, формує стиль одягу, стиль спілкування, стиль всього життя.

Різна музика має різний вплив на людину. Адже деякі мелодії здатні навіть покращувати людську пам'ять, вони також допомогають вибудовувати асоціативний ряд певних подій і моментів життя. Виявляється, що музика має величезний вплив на інтелектуальні здібності і можливості людини, адже мелодія музики сприяє збільшенню емоційної людської активності. І, як виявили, саме в цей період активності, підвищуються інтелектуальні здібності.

Дуже цікавий факт полягає в тому, що спокійна мелодія може позитивно впливати на кров. Та мелодія, яка викликає у людини задоволення, активно збільшує в крові обсяг лімфоцитів, і відповідно всьому організму людини стає набагато простіше боротися з будь-якими хворобами. Як виявили психологи, мелодія з ритмом 60 ударів на хвилину, діє на людину як медитація, вона цілком здатна відвернути будь-яку людину від будь-якої проблеми, сповільнюючи мозкову діяльність. Якщо слухати таку музику, активно поліпшується пам'ять, працездатність, спокій і впевненість у собі.

З розвитком мобільних технологій музика стала доступна для людей в цілодобовому режимі, а отже стала невід’ємною частиною життя суспільства.

Все частіше ми бачимо на вулиці, у громадському транспорті та ресторанах людей з наушниками за прослуховуванням музики. Музика стала настільки важливою в повсякденному житті, що винахідники змагалися за право першим винайти спосіб запису і збереження музики. В наш час це не є проблемою, музика доступна на багатьох платформах і веб-сайтах.

На сьогоднішній день ця галузь розвинена і представлена великим різноманіттям звукозаписуючої апаратури, а також мініатюрних систем, які вмонтовані в різні прилади, якими користується людина.

Великою проблемою є те, що з розвитком індустрії звукозапису і ростом кількості жанрів і стилів музики зникає можливість мати доступ до усього розмаїття музики від минулого до сьогодення, яке було б впорядковане за багатьма критеріями, чітко вказано жанр і піджанр музики, а також можливість зручного пошуку і відтворення музики на будь-якому пристрої,а також збереження її у разі потреби. Кожна сучасна система дає обмежене коло музичних творів, орієнтоване на аудиторію країн чи регіонів де вона буде розвиватися. Також більшість систем є платними і мають обмежений функціонал. Великою проблемою є те, що у більшості безкоштовних систем якість звуку залишає бажати кращого, оскільки є сильно стисненною для економного зберігання. А найбільшою проблемою є те, що більшість сучасних музичних сервісів мають низький рівень оптимізації, низьку швидкодію та надмірну кількість запитів, що при слабкому з’єднанні до мережі інтернет робить функціонування веб-сервісу занадто надлишковим.

При сучасному ритмі життя користувачі мають великий вибір веб-сервісів, тому питання оптимізації виведення, а саме реактивного виведення даних, є одним із найбільш важливих. Великим здобутком буде розробка методів реактивного виведення даних.

1.2 Порівняльний аналіз аналогів

Проблема реалізації реактивного виведення даних для веб-сервісів не є новою. Одним із методів, що дозволить здійснити реактивне виведення даних, є збереження даних до проміжного сховища. Цей метод також не є новим, оскільки існує декілька способів проміжного зберігання даних, яке дозволяє керувати роботою веб-сервісу, а також зменшити кількість запитів до серверу. Розрізняють два основних типи веб-сховища: локальне сховище (localStorage) і сесійне сховище (sessionStorage), що поводяться аналогічно постійним (Persistent cookie) і сесійним (Session cookie) кукам відповідно. Розглянемо кілька з них.

«localStorage» – локальне сховище для зберігання даних взаємодії користувача з веб-сервісом. Дані зберігаються за допомогою елементів (пар "ключ-значення"). Ключ являє собою ідентифікатор, який пов'язаний зі значенням. Тобто для того щоб записати або отримати необхідне значення порібно знати його ключ. Значення являє собою рядок, це необхідно враховувати при роботі з ним в коді JavaScript.

Мета – глобально зберігати дані робити користувача в браузері, що дозволить відновляти роботу веб-сервісу, якщо вікно браузеру чи сам браузер буде закрито.

Приклад зберігання даних в localStorage зображено на рисунку 1.1.

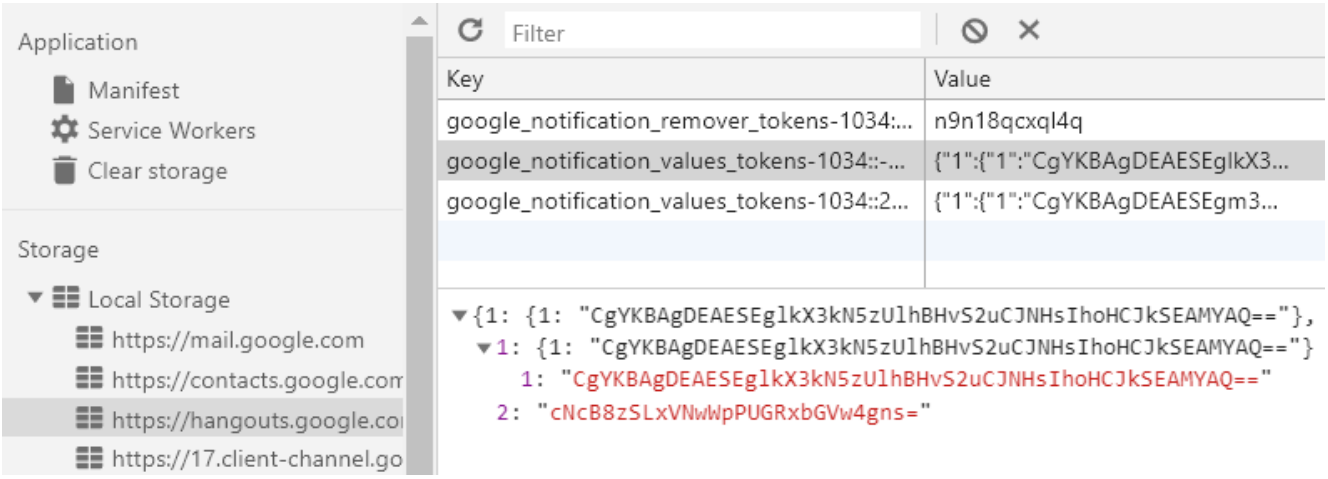


Рисунок 1.1 – Приклад зберігання даних в localStorage

«sessionStorage» — сесійного сховище для зберігання локальної сесії. Дія сесійного сховища обмежується життям даного вікна, тобто для кожного відкритого вікна створюється нова сесія, яка припиняє своє існування при закритті вікна. Збереження сесії призначене для надання окремих екземплярів одного і того ж веб-додатку для роботи в різних вікнах, не заважаючи один одному. У випадку з куками подібне зробити вкрай важко або навіть неможливо. Об'єкт sessionStorage зберігає дані обмежений час, вони видаляються відразу після того як користувач завершує свій сеанс або закриває браузер. Сесійне сховище підтримує набагато більше місця на диску в порівнянні з куками, яким доступно всього 4 Кб, що приблизно в 1000 разів менше ніж у сесійного сховища.

Приклад використання сесійного сховища зображено на рисунку 1.2.

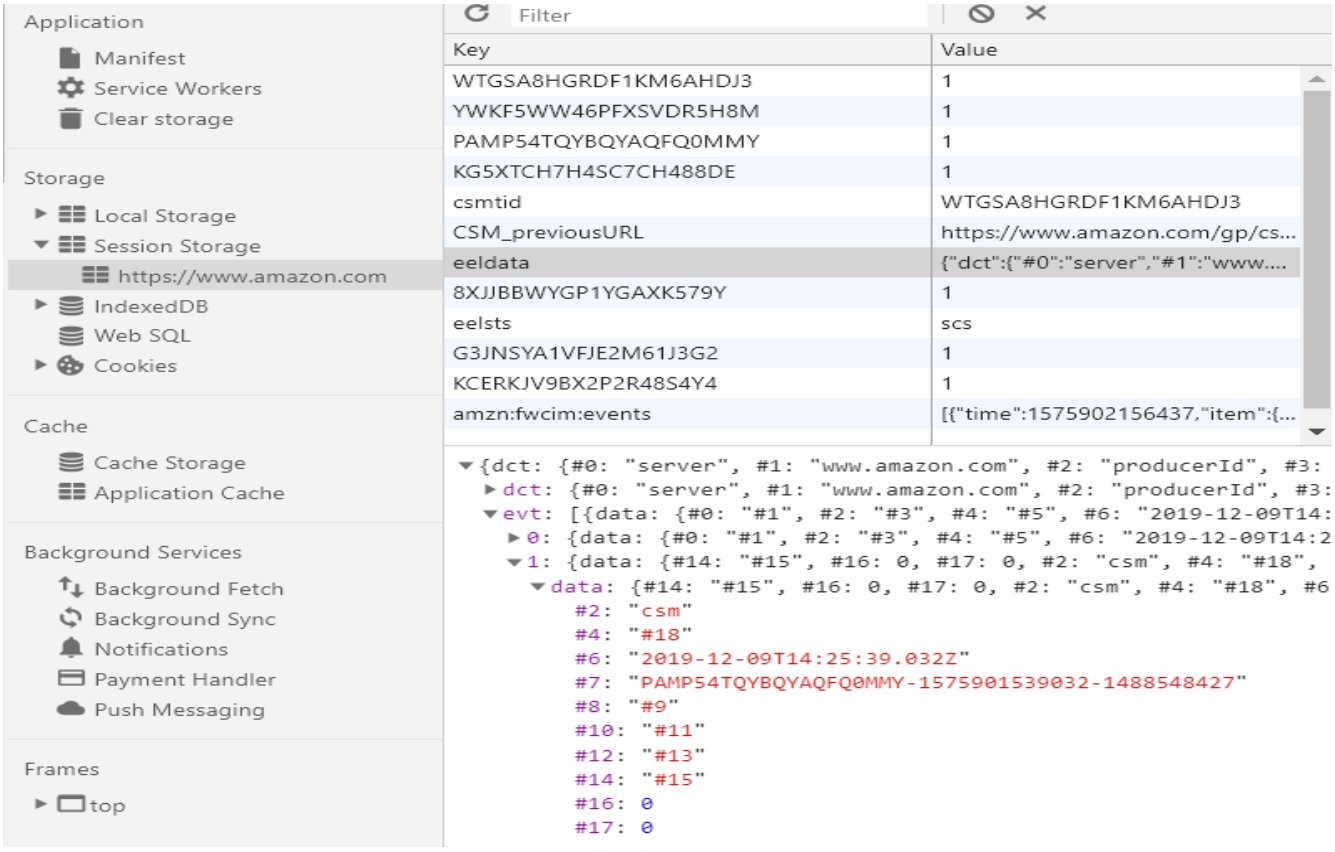


Рисунок 1.2 – Приклад використання сесійного сховища

«Файли cookie» — це невеликі текстові файли, які зберігаються в браузері Вашого комп'ютера або мобільного телефону після відвідування веб-сайтів. Файли cookie широко застосовуються для підтримки роботи веб-сайтів або для

підвищення якості досвіду користувача, а також для надання певної інформації власникам веб-сайту. Файли cookie, які використовують веб-сервіси можуть включати в себе інформацію про уподобання користувача в Інтернеті, щоб розробники веб-сервісу могли зробити продукт максимально цікавими для користувача. Файл «cookies» (невеликий файл з налаштуваннями профілів) полегшує користування веб-сайтом, записуючи дані, необхідні для входу в систему та збору статистики. Користуватися сайтом можна також без файлів «cookies». Куки зазвичай зберігають ваші вподобання на сайті, такі як ваша мова, місце перебування, тощо. Існують такі види куки-файлів: «сеансові» і «постійні». Сеансові куки є тимчасовими і зберігаються тільки до моменту вимкнення браузера. Постійні куки залишаються на жорсткому диску комп'ютера або на електронному носії до тих пір, поки їх не видалити, або не закінчиться термін їх дії.

Приклад зберігання cookie файлів зображено на рисунку 1.3.

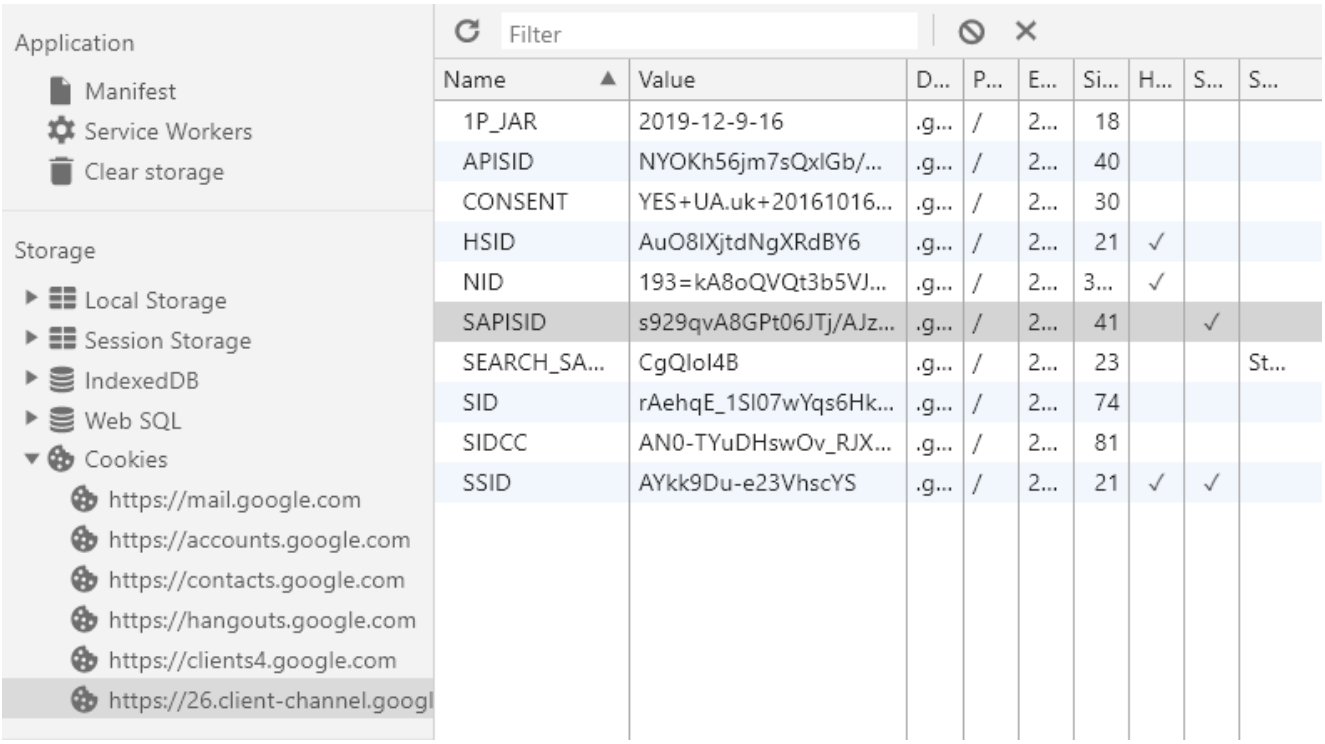


Рисунок 1.3 – Приклад зберігання cookie файлів

Розглянуто особливості функціонування кожного з методів локального збереження даних. Визначено, що кожен з цих методів має окремий підхід до

вирішення поставленої проблеми, а також свої недоліки. Вирішено, що потрібно об’єднати принципи і фунціонування кожного з методів в одне глобальне сховище даних(далі Store), а також додати можливість селективного вибору та реактивного виведення даних, зміна яких відслідковується автоматично[1].

Загальне порівняння аналогів та проміжного сховища Store наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння проміжного сховища «Store» з аналогами

	«Local Storage»	«Session Storage»	«Файли cookie»	«Store»
Довготривале зберігання даних	+ (1.0)	–	+ (1.0)	+ (1.0)
Підтримка складних і багаторівневих структур даних	+ (0.3)	+ (0.3)	+ (0.3)	+ (1.0)
Підтримка селективного вибору даних	–	–	–	+ (1.0)
Автоматичне відслідковування і оновлення даних	–	–	–	+ (1.0)
Потокове виведення даних в шаблон	–	–	–	+ (0.9)
Загальна сума	1.3	0.3	1.3	4.9

Проаналізувавши таблицю 1.1, а також підсумувавши коефіцієнти у кожному методі збереження і виведення даних можна зробити висновок, що найбільше значення «4.9» у проміжного сховища «Store», оскільки він поєднує

базовий набір функцій усіх розглянутих сховищ даних, в той же час розширюючи функціонування методу новими можливостями. Тому розробка методів реактивного виведення даних, включаючи проміжне сховище даних «Store», є доцільною.

1.3 Постановка задачі

Методи реактивного виведення даних призначені для зручного зберігання отриманих даних, селективного вибору і автоматичного відслідковування змін даних, а також потокового виведення даних в шаблон.

Розглядаємо умовний функціонал, щоб визначити задачі по реалізації методів реактивного виведення даних.

При запуску веб-сервісу відправляється запит до серверу на отримання даних для початкової сторінкі, таких як: список популярних пісень, жанри музики, категорії аудіофайлів, дані про вхід тощо. Одразу ж формується підписка на відповідь з серверу і так званий «спостерігач» починає відслідковувати джерело даних[2]. В цей час відкривається головна сторінка на якій відображається анімація завантаження даних, очікуючи дані з серверу. Коли дані з серверу надходять, спостерігач запускає обробник результату запиту до серверу(успішне отримання чи помилка) і зберігає в проміжне сховище даних, в Store, у потрібне поле об’єкту. Метод селективного вибору відслідковує зміну стану сховища і відправляє нові зміни у компонент, в якому підписалися на отримання цих даних. Компонент бачить отриманні дані і реактивно виводить їх користувачу. При перезавантаженні чи закриті вкладки веб-сервісу, або ж завершенні життєво циклу конкретного компоменту, дані не буде втрачено і за новим звертанням користувача усі дані будуть завантаженні не з серверу, а реактивно виведено з проміжного сховища даних.

Отже, визначено основні задачі, які необхідно реалізувати у методах реактивного виведення даних:

1. Розробка проміжного сховища даних Store.

2. Надання можливості збереження багаторівневих структур даних.
3. Підтримка різних типів даних.
4. Підтримка потокового виведення даних.
5. Надання можливості селективного вибору даних.
6. Автоматичне відслідковування даних.
7. Можливість обробки відповіді з серверу.
8. Наявність початкового стану сховище даних.
9. Швидкий доступ до сховище з будь-якого компоненту.

Основні вимоги до методів реактивного виведення даних описано у технічному завданні у додатку А.

1.4 Висновки

Проаналізовано предметну область. Розглянуто методи вирішення проблеми реактивного виведення даних, що включають проблему реалізації багатофункціонального проміжного сховища даних, та 3 найбільш популярних проміжних сховищ даних.

Таким чином, виявлено проблеми і недоліки цих сховищ, які полягають у відсутності селективного вибору та потокового виведення даних. Також враховано, що методи реактивного виведення даних повинні підтримувати можливість збереження багаторівневих структур даних.

Досліджено, що в аналогах відсутня можливість автоматично відслідковувати зміни даних, а також оновлювати взаємопов’язані з ними дані. Існують інші методи, направлені на вирішення проблеми. Проте дані методи цілком не вирішують проблему, тому є необхідність у розробці методів реактивного виведення даних.

В результаті, виявлено доцільність розробки методів реактивного виведення даних, що включає розробку проміжного сховища даних Store.

Розроблено основні задачі та технічне завдання для методів реактивного виведення даних.

2 РОЗРОБКА АРХІТЕКТУРИ МЕТОДІВ РЕАКТИВНОГО ВИВЕДЕННЯ ДАНИХ

2.1 Розробка загальної архітектури методів

Для подальшого проектування методів реактивного виведення даних, є необхідність сформувати її загальну архітектуру за допомогою схематичного представлення кожного з методів, взаємодії об’єктів веб-сервісу, діаграми послідовності. Проектування схем і діаграм здійснюється на основі поставлених задач, описаному в розділі 1.3.

Архітектура методу збереження даних до проміжного сховища Store для веб-сервісу представлена на рисунку 2.1.

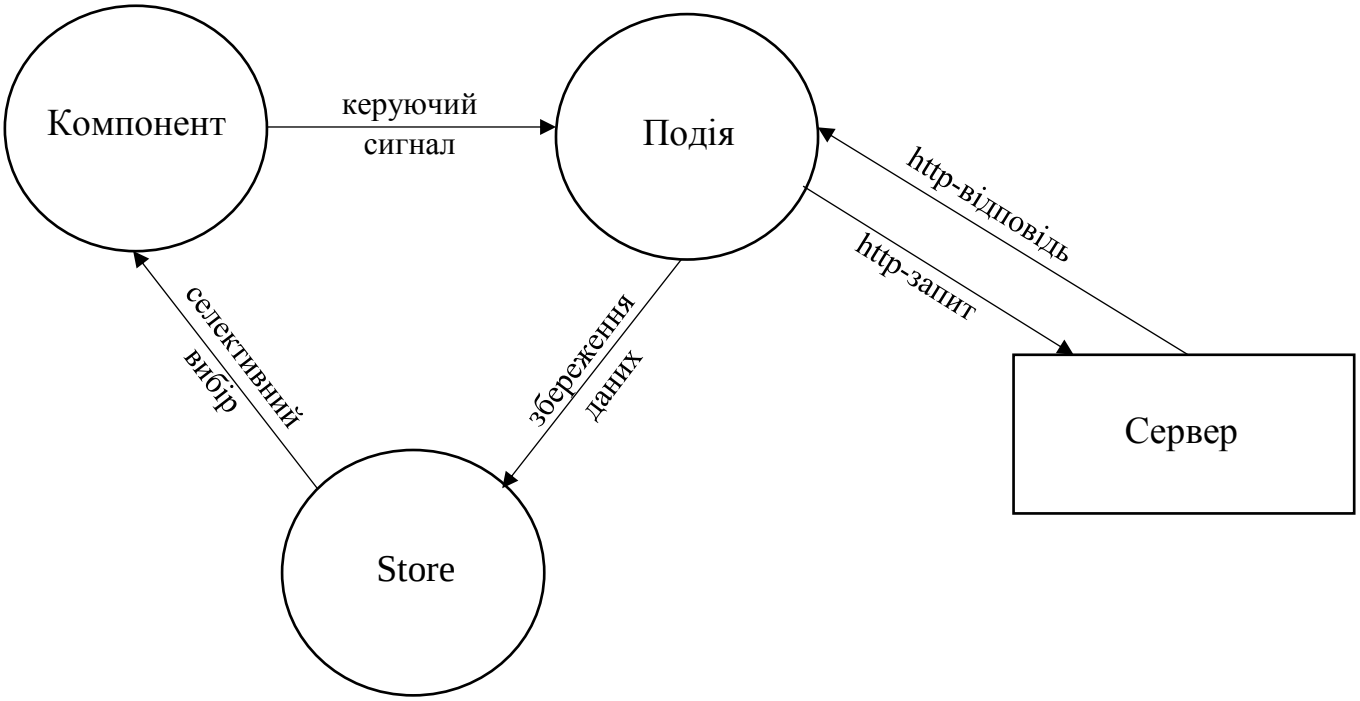


Рисунок 2.1 – Архітектура методу збереження даних до проміжного сховища Store

Оскільки розробка методів реактивного виведення даних проводиться для веб-сервісу зберігання та відтворення аудіофайлів, слід роглянути взаємодію об'єктів веб-сервісу.

Наявні 2 актори – система та користувач, які виконують властиві для них прецеденти. Особливістю є те, що система здійснює оброблення всіх даних, в залежності від вибору певних дій користувачем.

Взаємодія об'єктів веб-сервісу представлено на діаграмі послідовності на рисунку 2.2.

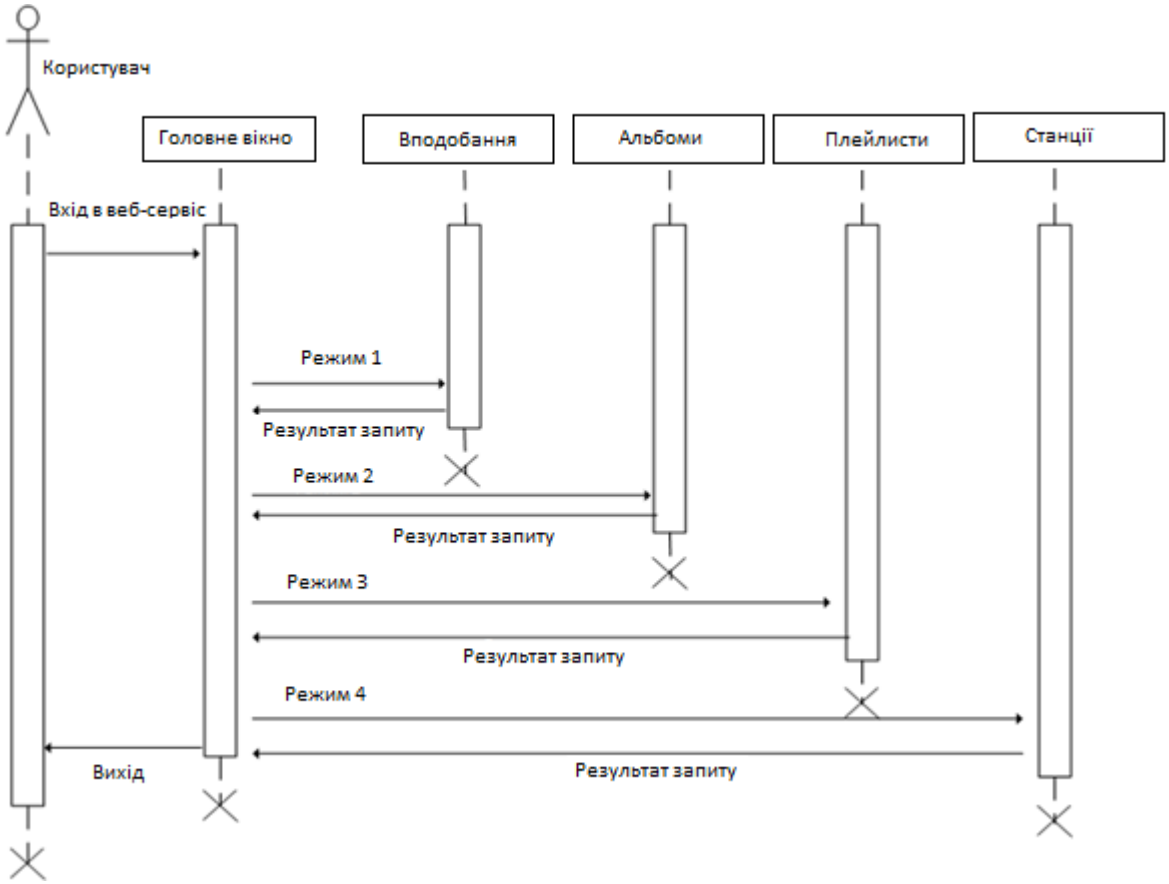


Рисунок 2.2 – Діаграма послідовності

Об'єктами веб-сервісу є «Користувач», «Головне вікно» та режими використання веб-сервісу, які може обрати користувач. Продемонстровано послідовність взаємодії користувача з додатком у часі. Спочатку користувач логується, щоб перейти в головне вікно додатку, а лише потім обирає бажаний режим використання веб-сервісу. Вибір режиму є рівноправним, тобто з головного вікна програми можна перейти в будь–який із режимів використання.

Структурна карта Константайна є моделлю відносин ієрархії між програмними модулями. Вона представляє собою взаємодію модулів додатку

«myMusic» і зображена на рисунку 2.3 у вигляді графу. Вершини – модулі, дуги – міжмодульні зв’язки. Головним модулем є модуль інтерфейс. Він пов’язаний зв’язком по управлінню з модулями візуалізації та модулем керування програми.

Також, відповідним зв’язком поєднується з локальними даними. Модуль керування пов’язаний зв’язком по данним з локальними даними, які включають в себе логін та пароль користувача, дані про вибір одного з модулів веб-сервісу. Модуль керування використовує локальні дані та на їх основі виконує одну з підсистем – режими використання веб-сервісу «myMusic» (вподобання, альбоми, плейлисти, станції) та функцію перевірки оновленнь аудіофайлів.

Кожна з підсистем взаємодіє з модулем візуалізації зв’язком по данних, оскільки відповідний модуль використовує дані як для відображення відповідних компонентів підсистем, так і дані результату їх виконання.

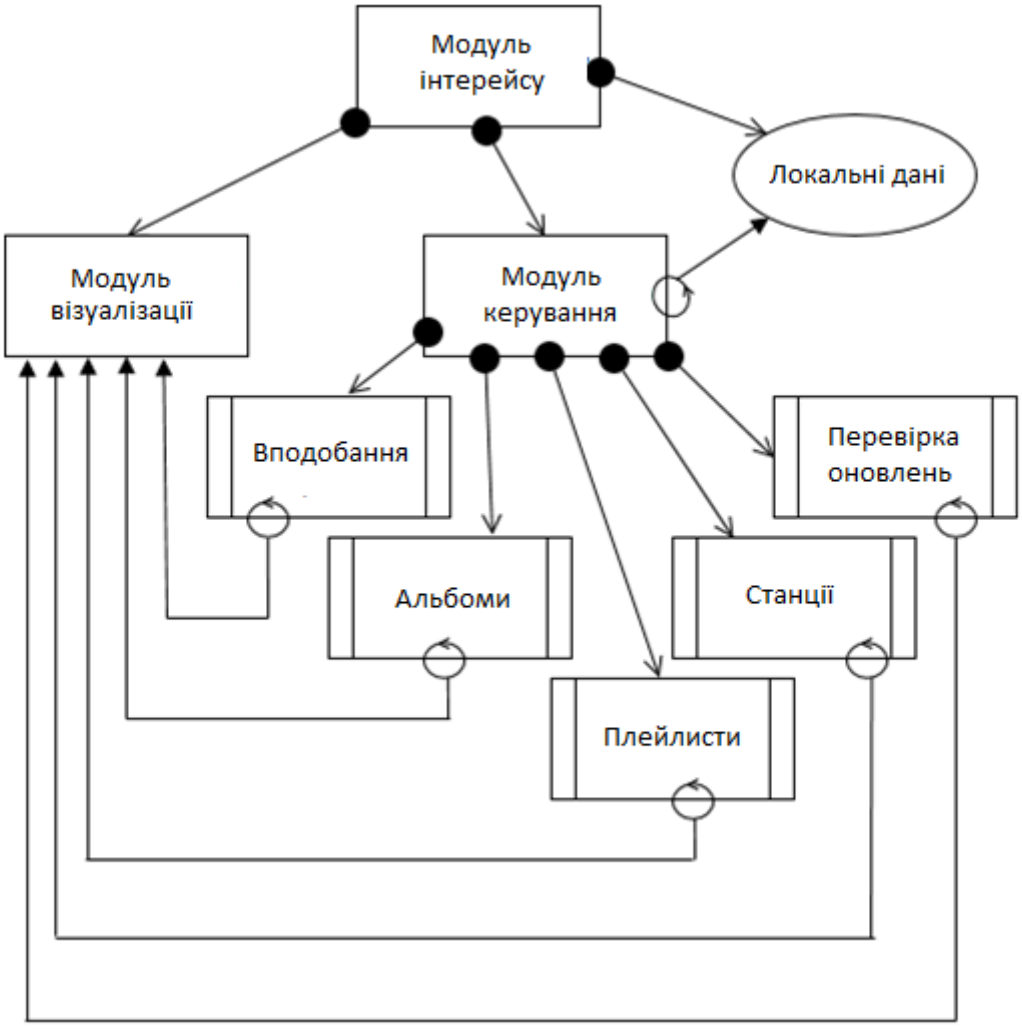


Рисунок 2.3 – Структурна карта Константайна веб-сервісу «myMusic»

Структурна схема компонентів веб-сервісу «myMusic» розглянуто на рисунку 2.4.

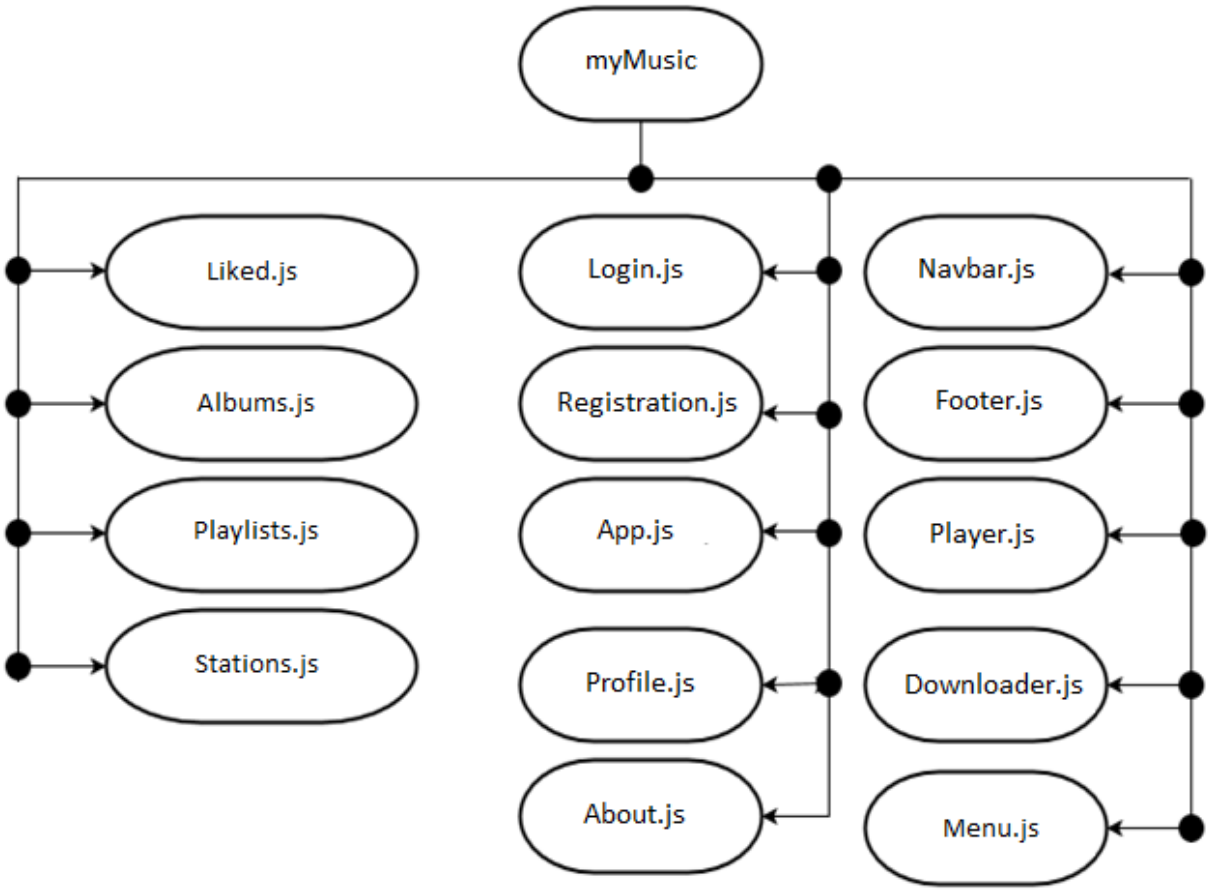


Рисунок 2.4 – Структурна схема компонентів веб-сервісу «myMusic»

Основними компонентами веб-сервісу «myMusic» є:

1. Login.js – компонент призначенний для входу в систему користувача.
2. Registration.js – компонент призначенний для реєстрації користувача в веб-сервісі.
3. App.js – головний модуль веб-сервісу, який забезпечує роботу всіх інших компонентів.
4. Profile.js – компонент призначенний для відображення профілю користувача, його даних.

5. About.js – компонент призначенний для отримання інформації про веб-сервіс, його можливості та розробників.

6. Navbar.js – компонент призначенний для відображення загаловку, шапки веб-сервісу, відповідає за розміщення логотипу веб-сервісу і слугує інструментом навігації.

7. Footer.js – компонент призначенний для відображення «підвалу» веб-сервісу, в ньому може розміщатися ім’я автора, дата останнього оновлення сервісу, а також контактна і правова інформація.

8. Player.js – компонент призначенний для відображення плеєру для відтворення аудіофайлів.

9. Downloader.js – компонент призначенний для завантаження власних аудіофайлів.

10. Menu.js – компонент призначенний для відображення головного меню веб-сервісу.

11. Liked.js – компонент призначенний для відображення вподобанних аудіофайлів користувача.

12. Albums.js – компонент призначенний для відображення створенних користувачем альбомів.

13. Playlists.js – компонент призначенний для відображення існуючих плейлистів користувача.

14. Stations.js – компонент призначенний для відображення аудіофайлів, що грунтуються і є подібними до останнього відтворенного файлу з альбому, плейлисту тощо.

2.2 Розробка методів виведення

Для розроблюваних методів реактивного виведення даних, згідно архітектури, що вказано в попередньому підрозділі, необхідно реалізувати потокове виведення даних, яке дозволятиме реактивно виводити дані в компонент, а також відслідковувати оновлення даних після відповіді серверу[3].

Архітектура методу потокового виведення даних для веб-сервісу представлена на рисунку 2.5:

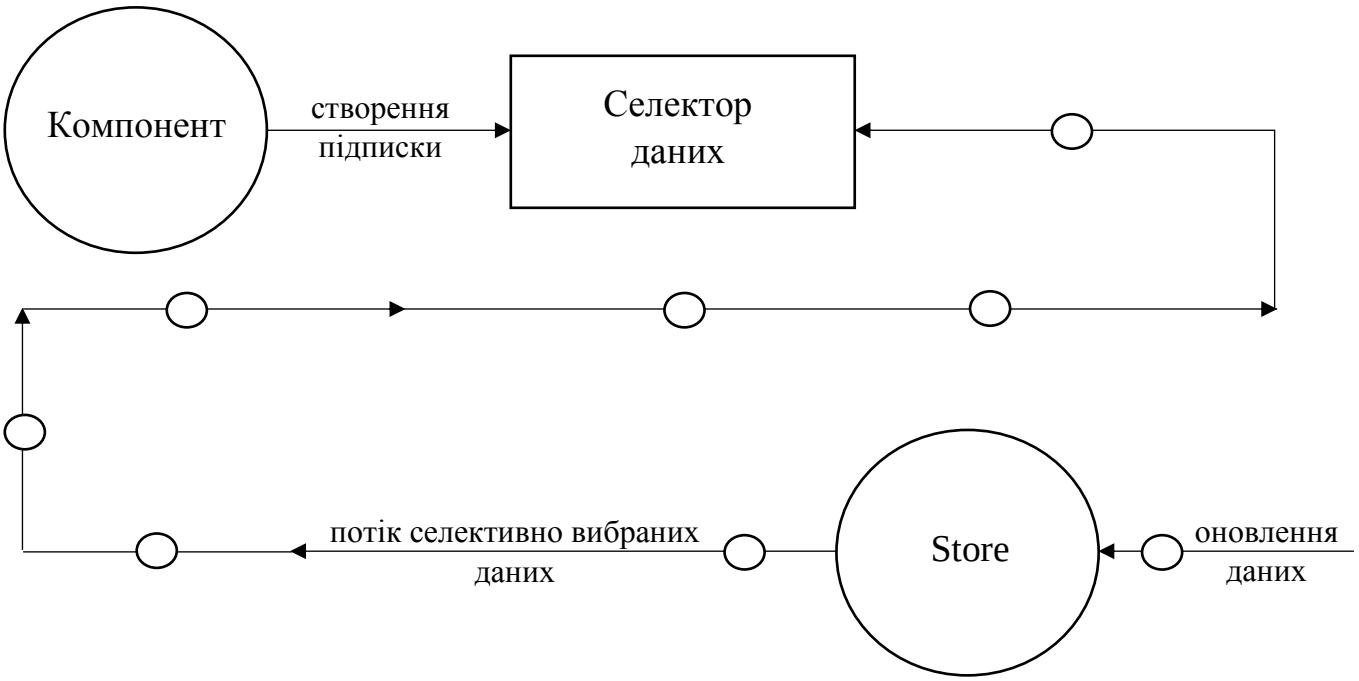


Рисунок 2.5 – Архітектура методу потокового виведення даних

В момент ініціалізації компонента для відображення користувачеві відправляється керуючий сигнал, що запускає подію, яка робить http-запит до серверу, якщо дані в сховищі відсутні.

В компоненті веб-сервісу створюється підписка на селектор даних зі сховища даних Store, яка відслідковує зміни в екземплярі сховища і при оновленні даних, отриманих після відповіді на http-запит, автоматично додає їх в потік на який підписався компонент.

Після отримання даних зі сховища завдяки потоковому виведенню, шаблон компоненту може відображати дані за допомогою раніше сформованої підписки на селектор даних.

Після програмної реалізації методів реактивного виведення даних необхідно створити підписки на селектори даних в таких компонентах:

- головний компонент веб-сервісу;
- компонент логування;

- компонент реєстрації;
- компонент шапки;
- компонент підвалу;
- компонент сторінки користувача;
- компонент програвача аудіофайлів;
- компонент завантажувача аудіофайлів;
- компонент меню;
- компонент довідки;

Для реалізації потокового виведення даних найкраще підійде потужний інструмент реактивного підходу такий як RxJs, реалізація парадигми реактивного програмування для мови програмування JavaScript.

Реактивне програмування – це програмування з асинхронними потоками (streaming) даних. Інакше кажучи, реактивність – це здатність миттєво реагувати на будь-які зміни, в першу чергу це стосується реагування на зміну даних та орієнтацію на потоки. Якщо порівнювати два підходи до програмування, то на відміну від імперативного підходу, реактивний підхід будується на «push» стратегії поширення змін. «Push» стратегія реалізовує те, що в разі зміни даних ці самі зміни будуть «проштовхуватися», і залежні від них дані будуть автоматично оновлюватися[3].

Справа з асинхронною неблокуючою обробкою завжди була присутня в мові програмування JavaScript, і зараз стає дуже популярною у багатьох інших контекстах.

Переваги зрозумілі: ефективне використання ресурсів, покращення швидкодії. Але вигоди приносять свою ціну: нетривіальне збільшення складності. Реактивне програмування є складною за своєю природою, наш розум звик мислити послідовно.

В свою чергу, RxJS - це бібліотека для JS, яка використовує патерн Observable (з англ. "Спостерігач") для спрощення обробки і компоновки асинхронного або callback коду. Бібліотека надає основний тип Observable та

кілька допоміжних типів (Observer, Schedulers, Subjects) і оператори роботи з подіями як з колекціями (map, filter, reduce, every і подібні з JavaScript Array)[4].

Потік – це масив даних, відсортованих за часом, який може повідомляти про те, що дані змінилися. І все, що необхідно зробити, це підписатися на потік і він автоматично повідомить коли відбудуться зміни. І це вміє робити бібліотека RxJs.

Приклад використання потіку в реактивному програмуванні зображено на рисунку 2.6.

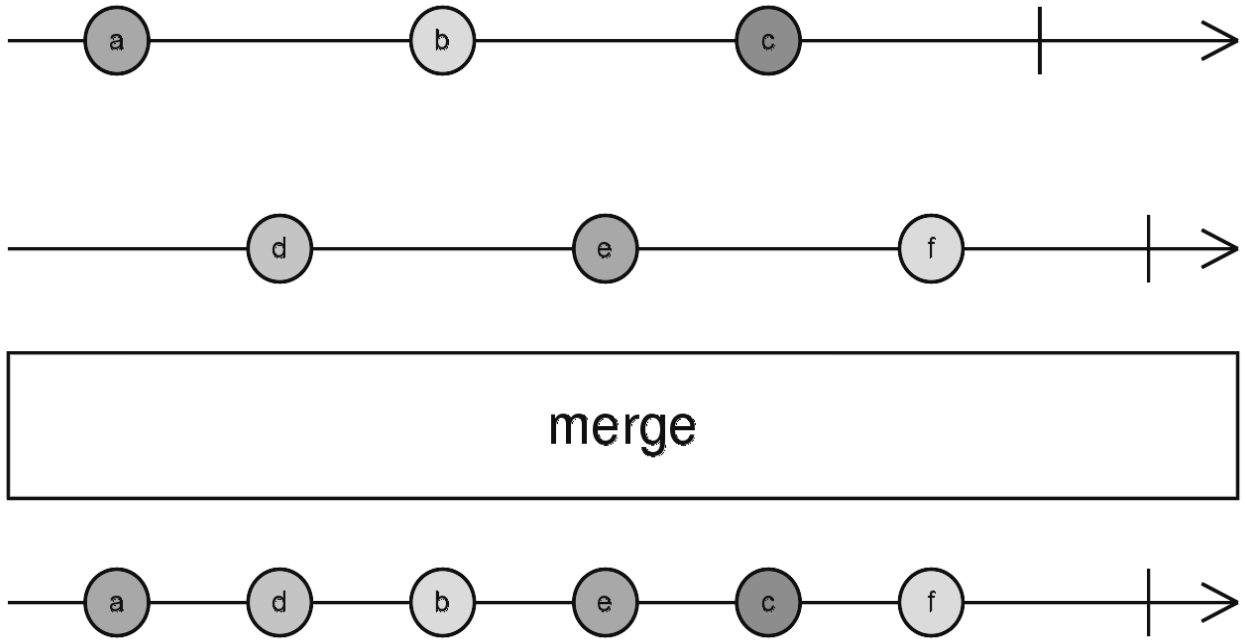


Рисунок 2.6 – Злиття двох потоків в RxJs

2.3 Проектування глобального сховища даних

Для зберігання інформації від користувача буде використана база даних NoSQL «MongoDB». MongoDB — документно-орієнтована система керування базами даних (СКБД) з відкритим серцевим кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журнальні операцій зі зміни і додавання даних в БД.

Всю модель пристрою бази даних в MongoDB можна представити таким чином (рис. 2.7):

Рисунок 2.7 – Модель бази даних в MongoDB

Для реалізації збереження повідомлень користувачів та необхідних даних з веб-сервісу для зберігання і відтворення аудіофайлів в базу даних, було розроблено структурну схему розробленої бази даних (рис. 2.8). На ній зображено колекції, які будуть використанні в веб-сервісі для зберігання і відтворення аудіофайлів. Можна виділити 3 основні колекції, такі як: «Користувачі», «Аудіофайли», «Аудіофайли користувача».

Колекція «Користувачі» слугуватиме для збереження інформації про користувача, його «Логін» та зашифрований «Пароль», а також масив з інформацією профілю, такою як «Фото», «Ім’я», «Прізвище», «Кількість слухачів».

Колекція «Аудіофайли» слугуватиме для збереження інформації про аудіофайл, а саме: «Назва» , «Виконавець», «Обкладинка», «Дата додання», «Кількість вподобанн».

Колекція «Аудіофайли користувача» слугуватиме для збереження інформації про аудіофайли, які вподобав, додав в альбом, плейлист користувач, а саме: «Ідентифікатор користувача» – вказуватиме на конкретного користувача веб-сервісу, «Ідентифікатор аудіофайлу» – вказуватиме на конкретний аудіофайл обраний або занатажений користувачем, «Категорія аудіофайлу» – вказуватиме на певну категорію до якої відноситься аудіофайл користувача (вподобання, альбом чи плейлист користувача).

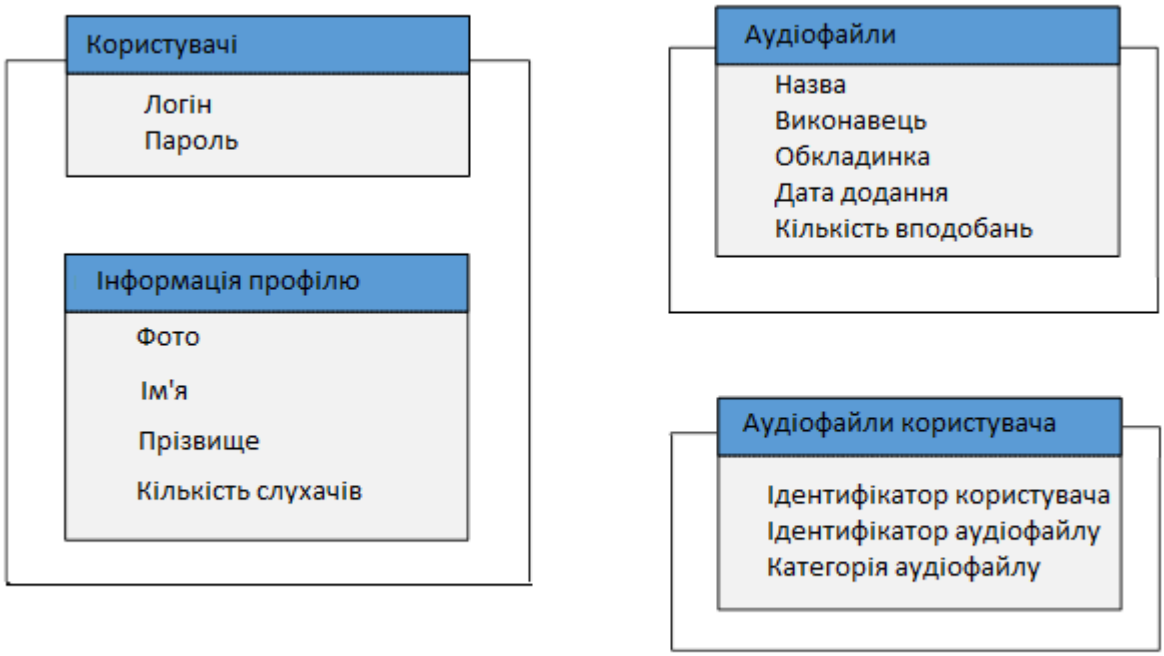


Рисунок 2.8 – Структурна схема бази даних для зберігання та відтворення аудіофайлів

За аналогічною структурою буде проводитись зберігання до проміжного сховища даних Store, оскільки потрібно дотримуватися відповідності між отриманими даними з сервера та збереженими до проміжного сховища структурами, щоб уникнути мутації даних. Єдиним виключенням буде пароль користувача, оскільки збереження в локальному сховищі таких даних може призвести до втрати ваших персональних даних.

2.4 Вибір технічних засобів

Підчас розробки методів реактивного виведення даних було обрано декілька технічних засобів, які значно полегшили процес розробки методів. Технічні засоби предсавлені набором утиліт, плагінів, бібліотек.

ReduxDevTools – це плагін, розширення для браузеру, яке відображає структуру збереженного стану веб-сервісу, а саме: інформація про поточного користувача, завантаженні аудіофайли, список відображених аудіо, рівень доступу, а загалом все, що потрібно користувачеві для швидкої роботи з веб-сервісом.

Розширення надає додаткові джерела живлення для вашого робочого процесу Redux. Крім Redux, він може використовуватися з будь-якою іншою архітектурою, яка керує станом додатку.

NGX Logger – це плагін, простий модуль реєстрації журналів для Angular (на даний момент підтримує Angular 6+). Плагін дозволяє "симпатичний друк" в консолі браузеру, а також дозволяє надсилати повідомлення журналу до URL-адреси для реєстрації на стороні сервера. Дозволяє виводити в консоль дані, що записуються в сховище даних. Полегшує тестування і розробку веб-сервісу. Показує минуле і поточне значення Store.

NGXS – є шаблоном керування станом та бібліотекою для Angular. Він діє як єдине джерело істинного стану вашої програми, забезпечуючи прості правила для передбачуваних мутацій стану[5].

NGXS спроектований після того, як шаблон CQRS, що широко застосовується в бібліотеках, таких як Redux і tgRx, знизив рівень boilerplate (шаблонний код) за допомогою сучасних функцій TypeScript, таких як класи і декоратори. CQRS pattern дозволяє швидко і зручно розробляти плагіни для NGXS (storage-plugin, dev-tools plugin, logger plugin тощо).

Переваги NGXS:

1. Використання того ж патерну як у NGRX.

2. Зниження рівня шаблонного коду завдяки використанню сучасних функцій TypeScript, таких як класи і декоратори.
3. Немає оператора перемикання.
4. Покращено читабельність коду.

Управління станом – це підхід, який передбачає управління одним або декількома елементами користувацького інтерфейсу, такими як текстові поля, кнопки, графічні компоненти та інші в GUI.

2.5 Висновки

В результаті, розроблено загальну архітектуру методів реактивного виведення даних за допомогою схем архітектури методів, відповідних діаграм послідовності, структурної карти Константайна, структурної схеми компонентів додатку. Описано призначення кожного з основних компонентів веб-сервісу. Кожна з діаграм розглядає методи реактивного виведення даних та веб-сервіс «myMusic» як цілісну систему, та описує різні аспекти методів реактивного виведення даних музичного веб-сервісу. Також спроектовано сховище даних, що призначене для зберігання даних про користувача і його аудіофайли. Вона необхідна в подальшому для фукціонування веб-сервісу, реалізацію запланованого функціоналу. Розроблено метод потокового виведення даних, описано його архітектуру і роботу. Описано призначення кожної колекції глобального сховища даних MongoDB та визначенно поля, необхідні для раціонального функціонування веб-сервісу. Обрано основні технічні засоби для реалізації методів реактивного виведення даних для веб-сервісу зберігання та відтворення аудіофайлів, доведено доцільність вибору і представленно кольори як в графічному, так і в кодовому вигляді. Відповідні матеріали будуть використані при програмній реалізації методів.

3 РОЗРОБКА МЕТОДІВ РЕАКТИВНОГО ВИВЕДЕННЯ ДАНИХ

3.1 Варіантний аналіз та обґрунтування засобів реалізації

В сфері веб-технологій процес розробки веб-додатків прийнято розподіляти на наступні етапи:

1. back-end development (розробка серверної частини веб-сайту);
2. front-end development (розробка клієнтської частини веб-сайту);
3. testing (тестування).

В подальшому відбувається процес Deployment (розгортання додатку на сервері) та Maintenance (супровід). Стек технологій, які слід використовувати, перш за все базується на основі вибору технологій та мови програмування для серверної і клієнтської частини сайту. На сьогоднішній день найпоширенішими серверними технологіями є:

1. PHP (Laravel, YII, Zend Framework, Symfony, CakePHP);
2. Java (Spring);
3. C# (.NET);
4. Ruby (Ruby on Rails);
5. Python (Django, Flask);
6. JavaScript (Node.js).

Розглянемо фреймворки для мови програмування JavaScript.

Node.js створений у 2009 році Райаном Далем. Він створив програмну платформу, засновану на JavaScript V8. Незвичним є те, що платформа має вбудовані бібліотеки для обробки запитів і відповідей, не потрібно використовувати сторонній веб-сервер і будь-які інші залежності. Node.js набирає обертів і використовується такими компаніями, як Microsoft, Yahoo, LinkedIn і PayPal[6].

Node.js характеризується такими властивостями[7]:

- асинхронна однопотокова модель виконання запитів;
- неблокуючий ввід/вивід;

- система модулів CommonJS;
- рушій JavaScript Google V8.

Для керування модулями використовується пакетний менеджер npm (node package manager), що є найголовнішою перевагою, тому що npm є найбільшою у світі екосистемою бібліотек з відкритим кодом.

Для керування модулями використовується пакетний менеджер npm (node package manager), що є найголовнішою перевагою, тому що npm є найбільшою у світі екосистемою бібліотек з відкритим кодом.

Одними з найбільш поширених фреймворків JavaScript є Angular, React та Vue.js.

Angular представляє фреймворк від компанії Google для створення клієнтських додатків. Перш за все він націлений на розробку SPA-рішень (Single Page Application), тобто односторінкових додатків.

Angular часто називають MVW (Model-View-Whatever) фреймворк і серед головних переваг для стартапів є: швидке написання коду, швидке тестування будь-якої частини програми та двостороння прив'язка даних (зміни в back-end відразу ж відображаються на призначеному для користувача інтерфейсі).

Переваги Angular:

- Підтримка веб-компонентів. Веб-компоненти Angular засновані на новому стандарті веб-компонентів, на відміну від закритої системи модулярізації AngularJS. На практиці це означає, що AngularJS дає можливість безпосередньо використовувати будь-який компонент, написаний як Web Component, не вдаючись до коду верстки.
- Використання Typescript. Найбільша комерційна перевага TypeScript полягає в його інструментарії. Ця мова дає можливості сучасного автозаповнення, навігації та рефакторінга. Такі інструменти стають практично незамінними при роботі з великими проектами.
- Відмінна продуктивність Angular не проводить глибокий порівняльний аналіз об'єктів. Якщо якийсь елемент додати в масив даних, зміну

шляху не буде виявлено. Це стосується і властивостей об'єкта, поки вони не пов'язані безпосередньо з View[8].

React (старі назви: React.js, ReactJS) — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників[9].

Основна роль React – бібліотека для створення користувацьких інтерфейсів. Він фокусується на «View» частини MVC-розробки й дозволяє створювати компоненти інтерфейсу, що зберігають свій стан. Це була одна з перших бібліотек, що реалізують віртуальне DOM-дерево. Статистика використання React може здатися досить низькою через те, що він використовується в основному в додатках, а не на сайтах[10].

Vue.js – JavaScript-фреймворк що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних[10].

Vue.js реалізовує двосторонню прив'язку даних (як в AngularJS), візуалізацію на стороні сервера (як в Angular2 і ReactJS), Vue-cli (каркасний інструмент для швидкого старту) і опціональну підтримку JSX. Його творець стверджує, що Vue2 є одним із найшвидших фреймворків в цілому. Vue.js — це найкращий вибір для швидкої розробки крос-платформних додатків. Він може стати міцною основою для високопродуктивних додатків в одну сторінку (SPAs) і вигідним рішенням для тих випадків, коли продуктивність важливіше, ніж хороша організація коду або структура додатку.

Кожен із розглянутих фреймворків є досить потужним та має свої недоліки та переваги. Порівняльний аналіз був проведений із оцінюванням використання цих засобів для створення мережевого журналу, що дає можливість обміну персональними даними, зображеннями або мультимедійними файлами[11].

Vue.js є легким сучасним фреймворком, але є досить молодим проектом, що має менше ресурсів ніж в альтернативних варіантів, тому його використання

є великим ризиком. React — це найбільш популярний фреймворк з найшвидшим розвитком, але складний для вивчення. React побудований на Javascript ES6, тоді як Vue – на ES5 або ES6, а Angular – на TypeScript. Використання в Angular TypeScript сприяло до підвищення надійності та безпеки, які необхідні для корпоративних додатків[12].

Перевагою у сторону Angular є також те, що найбільш популярна бібліотека RxJS, що використовує реактивний підхід, була взята за основу цього фреймворка.

JetBrains WebStorm — інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей [13]. WebStorm постачається з наперед встановленими плагінами JavaScript (такими як для Node.js). WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart.

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) — в таких конструкціях підтримується коректний рефакторинг .

3.2 Аналіз вибору мови програмування

Для реалізації методів потрібно обрати як мову програмування, так і середовище, в якому розроблятиметься додаток.

Розглядаємо мови розробки JavaScript, PHP, TypeScript.

JavaScript (JS) — динамічна, прототипно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина

браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки[11].

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

1. написання сценаріїв веб-сторінок для надання їм інтерактивності;
2. створення односторінкових веб-застосунків (ReactJS, AngularJS, Vue.js);
3. програмування на стороні сервера (Node.js);
4. стаціонарних застосунків (Electron, NW.js);
5. мобільних застосунків (React Native, Cordova);
6. сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
7. всередині PDF-документів тощо.

Переваги Javascript:

1. підтримують усі браузери;
2. підтримують такі програми: Adobe Photoshop, Adobe Illustrator, Adobe InDesign та Adobe Dreamweaver;
3. перевірка реєстраційних форм на помилки ще до відправлення на сервер;
4. створює яскраві та інтерактивні сторінки сайту;
5. можна здійснювати різного типу обчислення.

Мова стала популярнішою серед програмістів після появи AJAX технології, що створила новий етап у розробці сайтів. Тому відповідно вважається, що JavaScript є найпопулярнішою клієнтською мовою[12].

PHP (англ. PHP: Hypertext Preprocessor — PHP: гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проект відкритого програмного забезпечення.

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта.

PHP — мова, у код якої можна вбудовувати безпосередньо html-код сторінок, які, у свою чергу, коректно оброблюватимуться PHP-інтерпретатором. Обробник PHP просто починає виконувати код після відкриваючого тегу (<?php) і продовжує виконання до того моменту, поки не зустріне закриваючий тег (?>).

Велика різноманітність функцій PHP дає можливість уникати написання багаторядкових функцій, призначених для користувача, як це відбувається в C або Pascal.

TypeScript — мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки веб-застосунків, що розширює можливості JavaScript.

Розробником мови TypeScript є Андерс Гейлсберг (англ. Anders Hejlsberg), який створив раніше C#, Turbo Pascal і Delphi.

Код експериментального компілятора, котрий транслює код TypeScript в представлення JavaScript, поширюється під ліцензією Apache, розробка ведеться в публічному репозиторії через сервіс CodePlex.

TypeScript є зворотньо сумісним з JavaScript. Фактично, після компіляції програму на TypeScript можна виконувати в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js.

Переваги над JavaScript:

1. можливість явного визначення типів (статична типізація);
2. підтримка використання повноцінних класів (як в традиційних об'єктно-орієнтованих мовах);
3. підтримка підключення модулів.

За задумом ці нововведення мають підвищити швидкість розробки, прочитність, рефакторинг і повторне використання коду, здійснювати пошук помилок на етапі розробки та компіляції, а також швидкодію програм.

Порівняння мов програмування наведено в таблиці 3.1.

Таблиця 3.1 Порівняння мов програмування

	JavaScript	TypeScript	PHP
Швидкодія	+ (0.9)	+ (0.9)	+ (0.8)
Підтримка об’єктно–орієнтованого програмування	+ (0.7)	+ (0.7)	+ (0.7)
Підтримка структурного програмування	+ (0.6)	+ (0.6)	+ (0.6)
Простота та зручність розробленого коду	+ (0.9)	+ (0.7)	+ (0.6)
Підтримка типізації	+ (0.3)	+ (0.9)	+ (0.5)
Сума	3.4	3.8	3.2

Проаналізувавши таблицю 3.1, а також, підсумувавши коефіцієнти у кожній мові програмування, можна зробити висновок, що найбільше значення «3.8» у TypeScript. Тому вирішено, що мовою програмування для розробки музичного веб-сервісу «myMusic» буде TypeScript, оскільки надає всі необхідні засоби для швидкої реалізації методів.

3.3 Вибір середовища програмування

Розглянемо середовища для розробки веб-сервісів – JetBrains WebStorm, PhpStorm, Eclipse, Aptana Studio.

JetBrains WebStorm

JetBrains WebStorm — інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. WebStorm постачається з наперед встановленими плагінами JavaScript (такими як для Node.js).

WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart.

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями.

Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) — в таких конструкціях підтримується коректний рефакторинг [13].

PhpStorm

JetBrains PhpStorm — комерційне крос-платформове інтегроване середовище розробки для PHP, яке розробляється компанією JetBrains на основі платформи IntelliJ IDEA.

PhpStorm являє собою інтелектуальний редактор для PHP, HTML і JavaScript з можливостями аналізу коду на льоту, запобігання помилок у сирцевому коді і автоматизованими засобами рефакторинга для PHP і JavaScript. Автодоповнення коду в PhpStorm підтримує специфікацію PHP 5.3/5.4/5.5/5.6/7.0/7.1 (сучасні і традиційні проекти), включаючи генератори, співпрограми, простори імен, замикання, типажі і синтаксис коротких масивів. Присутній повноцінний SQL-редактор з можливістю редагування отриманих результатів запитів.

PhpStorm розроблений на основі платформи IntelliJ IDEA, написаної на Java. Користувачі можуть розширити функціональність середовища розробки за рахунок установки плагінів, розроблених для платформи IntelliJ, або написавши власні плагіни[14].

Робота з JavaScript, CSS і HTML:

1. Вся функціональність, доступна в WebStorm , включена в PhpStorm.
2. Автодоповнення коду для JavaScript, HTML і CSS (для тегів, ключових слів, міток, змінних, параметрів і функцій).
3. Підтримка HTML5
4. Live Edit: зміни в коді можна миттєво переглянути в браузері без перезавантаження сторінки
5. Підтримка CSS/SASS/SCSS/LESS (автодоповнення коду, підсвічування помилок, валідація тощо)
6. Zen Coding
7. Навігація по коду і пошук використань (перейти до оголошення / ідентифікатора, знайти використання)
8. Підтримка ECMAScript Harmony
9. Рефакторинг для JavaScript (перейменування, виділення змінної / функції, вбудовування змінної / функції, переміщення / копіювання, безпечне вилучення, витяг вбудованого скрипта в окремий файл)
10. Зневаджувач JavaScript, а також інтеграція з фреймворками модульного тестування JavaScript

Eclipse

Eclipse — вільне універсальне модульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, набір інструментів для програмістів на мові Java, системи контролю версій, конструктори GUI тощо. Написаний в основному на Java, може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах програмування, включаючи Ada, C, C++, COBOL, Fortran, Perl, PHP, Python, R, Ruby (включно з каркасом Ruby on Rails), Scala, Clojure та Scheme. Середовища розробки зокрема включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP[15].

Початок коду йде від IBM VisualAge, він був розрахований на розробників Java, складаючи Java Development Tools (JDT). Але користувачі могли розширяти можливості, встановлюючи написані для програмного каркасу Eclipse плагіни, такі як інструменти розробки під інші мови програмування, і могли писати і вносити свої власні плагіни і модулі.

Студія Aptana

Aptana Studio - інша широковикористовувана IDE для розробки веб-сайтів. Вона доступна або як окремий додаток, або як плагін для Eclipse. Aptana (компанія) також пропонує хостинг для ваших проектів, тому, якщо ви користуєтесь цією службою, IDE зв'язків у цих сервісах ідеально.

Є велика кількість інструментальних засобів для розробки веб-сервісів, кожен з яких має свої особливості. Вибір засобу залежить від того, які саме задачі потрібно вирішити при створенні додатку. Для розробки обрано JetBrains WebStorm, оскільки в даному засобі наявні всі необхідні засоби для розробки веб-сервісу «myMusic».

3.4 Розробка методу селективного вибору даних

Для реалізації методу селективного вибору даних спершу потрібно реалізувати методи керування станом проміжного сховища даних Store. Такі методи варто реалізувати окремо від розробленого компоненту. Для зручного зберігання даних варто розподілити сховище даних Store на функціонально незалежні об’єкти стану сховища(State), які мають визначену структуру даних, методи керування станом, а саме: події, селектори та початковий стан. Сховище даних Store може містити в собі багато станів. Для демонстрації розробки методів реактивного виведення даних, які включають метод селективного вибору даних, обрано базовий об’єкт Авторизації(AuthState) сховища даних Store.

Розглянемо розробку об’єкту Авторизації(AuthState) детальніше. Створення об’єкту Авторизації(AuthState) зображено на рисунку 3.1.

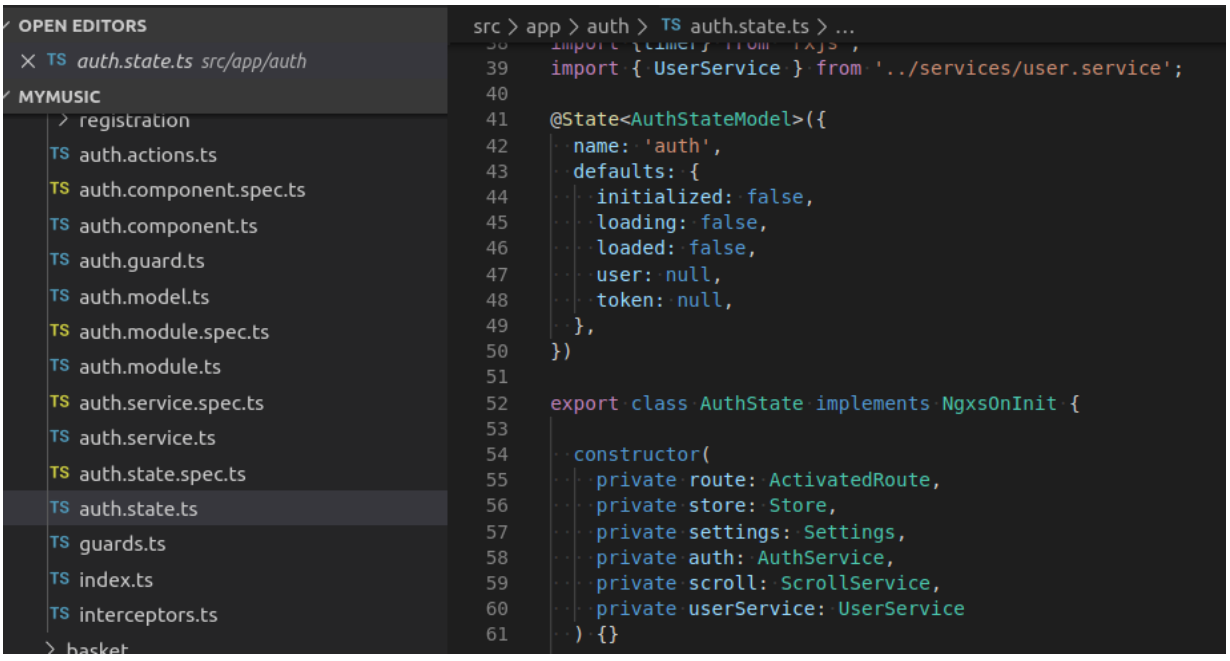


Рисунок 3.1 – Створення об’єкту Авторизації(AuthState)

Для створення об’єкту AuthState використовується декоратор @State з типом, що заданий інтерфейсом AuthStateModel. Поле defaults визначає потачковий стан об’єкту.

Структуру інтерфейсу AuthStateModel зображено на рисунку 3.2.

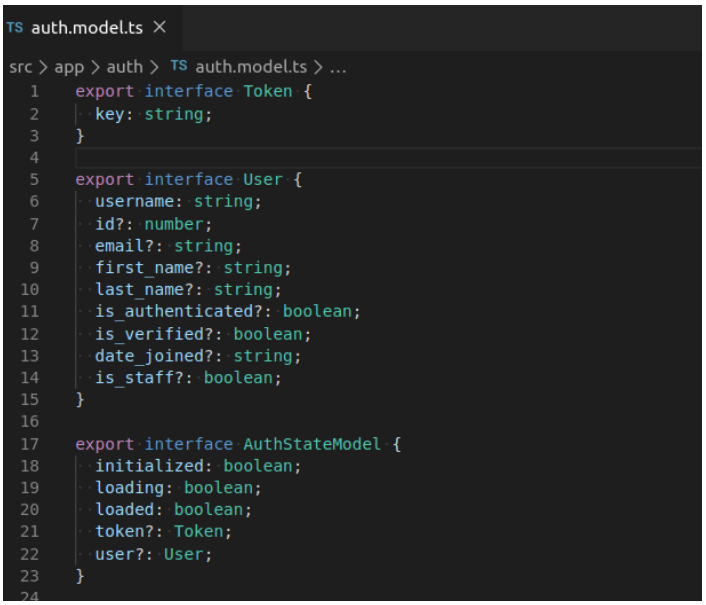


Рисунок 3.2 – Структуру інтерфейсів AuthStateModel, User та Token

Створення об’єкту AuthState, його методів та інтерфейсів реалізовано засобами NGXS та TypeScript. Метод селективного вибору даних реалізований за допопомогою декоратору @Selector. Реалізація методу селективного вибору даних для об’єкту AuthState зображено на рисунку 3.3.

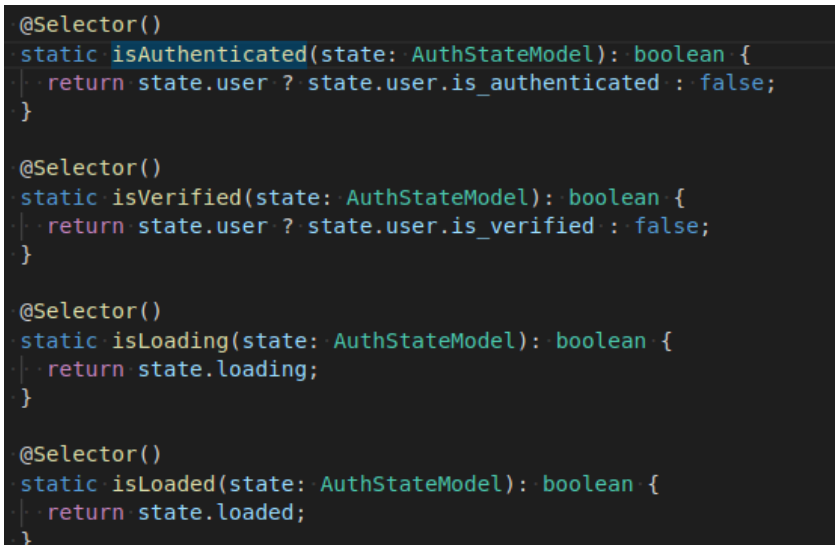


Рисунок 3.3 – Реалізація методу селективного вибору даних

Створення підписки на селектор даних зі сховища даних Store відбувається в компоненті і реалізовується за допомогою RxJS. Цей підхід дозволяє реалізувати потокове виведення даних зі сховища з використанням селективного

вибору, що є важливою частиною методу реактивного виведення даних. Створення підписки на селектор даних зображено на рисунку 3.4.

```
export class LoginComponent extends NgxsSingleFormComponent implements OnInit, OnDestroy {
  @Select(AuthState.isLoading) loading$: Observable<boolean>;
  @Select(MessagesState.getBackendError) backendErrors$: Observable<any>;

  constructor(
    protected store: Store,
    protected fb: FormBuilder,
    protected cd: ChangeDetectorRef,
    ) {
    super(fb, cd, store);
  }

  ngOnInit() {
    super.ngOnInit();
    this.store.dispatch(new Init());
  }

  protected performSubmit() {
    this.store.dispatch(new LoginWithEmailAndPassword(this.form.value));
  }
```

Рисунок 3.4 – Створення підписки на селектор даних об’єкту AuthState

Завдяки створенню підписки на селектор даних шаблон компоненту має доступ до потоку даних зі сховища, який надходить за допомогою селекторів, які в свою чергу відображають автоматично відслідковані дані з серверу. Використання потокового реактивного виведення в шаблоні компоненту зображено на рисунку 3.5.

```
th.model.ts    login.component.html ×
app > auth > login > login.component.html > div.content
<div class="content" [@loadContent]="'open'">
  <div class="text-center">
    <div class="text-center">
      If you have not created an account yet, then please
      <a routerLink="/auth/register/"><strong>Sign Up</strong></a> first.
    </div>
  </div>
  <div id="content_inner">
    <div class="row">
      <div class="col-md-6 col-md-offset-3 login_form">
        <form [formGroup]="form" class="well login" (ngSubmit)="performSubmit()">
          <h2>Log In</h2>
          <div class="form-group">
            <rt-input-block class="email" name='email' required='false' [label]="formLabels['email']" [form]="form"
              [errors]="formErrors"></rt-input-block>
          </div>
          <div class="form-group">
            <rt-input-block class="password" type="password" name='password' [label]="formLabels['password']" [form]="form"
              [errors]="formErrors"></rt-input-block>
          </div>
          <input type="hidden" name="login-redirect_url" id="id_login-redirect_url">
          <p><a class="forgotPassword" routerLink='/auth/password-reset'>I've forgotten my password</a></p>
          <button mat-flat-button color="primary" type="submit" [disabled]="loading$ | async">Log In</button>
        </form>
      </div>
    </div>
  </div>
</div>
```

Рисунок 3.5 – Потокове реактивне виведення в шаблоні компоненту

В даній реалізації реактивне виведення даних використовується для зміни властивості disabled кнопки Log In, відслідковуючи зміни сховища даних за допомогою підписки loading$ з компоненту та використання async pipe, що забезпечує виведення асинхронних даних в шаблоні.

3.5 Розробка візуальної частини

Для розробки візуальної частини програми (користувацького інтерфейсу), яка б відображала ефективну роботу методів реактивного виведення даних, було обрано використовувати HTML і CSS[16]. HTML (Hyper Text Markup Language) – стандартна мова розмітки, за допомогою якої виконується розробка структури веб-сторінок.

CSS (Cascading Style Sheets) – стандартна мова розмітки, за допомогою якої виконується розробка дизайну веб-сторінок.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

HTML5 — наступна версія мови HTML. До складу робочої групи з HTML5 увійшли AOL, Apple, Google, IBM, Microsoft, Mozilla, Nokia, Opera та кілька сотень інших виробників.

Існує деяка плутанина щодо версійності, оскільки існують дві незалежні групи розробників — WHATWG та W3C.

WHATWG відмовились від принципу «версійності» на користь «вічної розробки» при прийнятті специфікації HTML. Таке рішення було спричинено намаганням пришвидшити втілення стандарту в життя, тобто розробникам веб-браузерів не потрібно чекати допоки вийде офіційна затверджена версія специфікації (специфікація перейде в стан recomendation), вони можуть втілювати певні частини специфікації вже зараз.

У HTML5 включений елемент source для вказівки альтернативних відео і аудіо файлів, щоб браузер міг вибрати той, який підходить до підтримуваного медіа-типу або кодеків. Атрибут media визначає вибір медіа-запиту, що базується на обмеженнях пристроїв, а атрибут type — можливості медіа-типів і кодеків. Коли використовується атрибут source, слід опускати src в елементах video (audio), інакше source буде проігнорований.

Враховуючи всі переваги, прийнято рішення розробляти користувацький інтерфейс засобами HTML5 розмітки[17].

На рисунку 3.6 зображено процес створення користувацького інтерфейсу засобами HTML5 розмітки.

```
<section>
  <div>
    <ol class="row">
      <ul class="products" style="list-style: none">
        <li class="col-xs-11 col-sm-5 col-md-3 col-lg-3" [@loading]
          *ngFor="let item of (products$ | async); trackBy: productId">
          <app-product [item]="item"></app-product>
        </li>
      </ul>
    </ol>
    <div>
      <ul class="pager">
        <li class="next">
          <button *ngIf="next$ | async" class="btn btn-default showmore" (click)="showMore()">show
            more</button>
        </li>
      </ul>
    </div>
  </div>
</section>
</div>
</div>
</div>
```

Рисунок 3.6 – Створення користувацького інтерфейсу

Реактивне виведення є важливою частиною додатку, тому що добре розроблений інтерфейс гарантує швидку і зручну роботу користувача з програмою. На рисунках 3.7, 3.8 та 3.9 зображено розроблений користувацький інтерфейс для 3 сторінок програмного додатку, який використовує методи реактивного виведення даних.

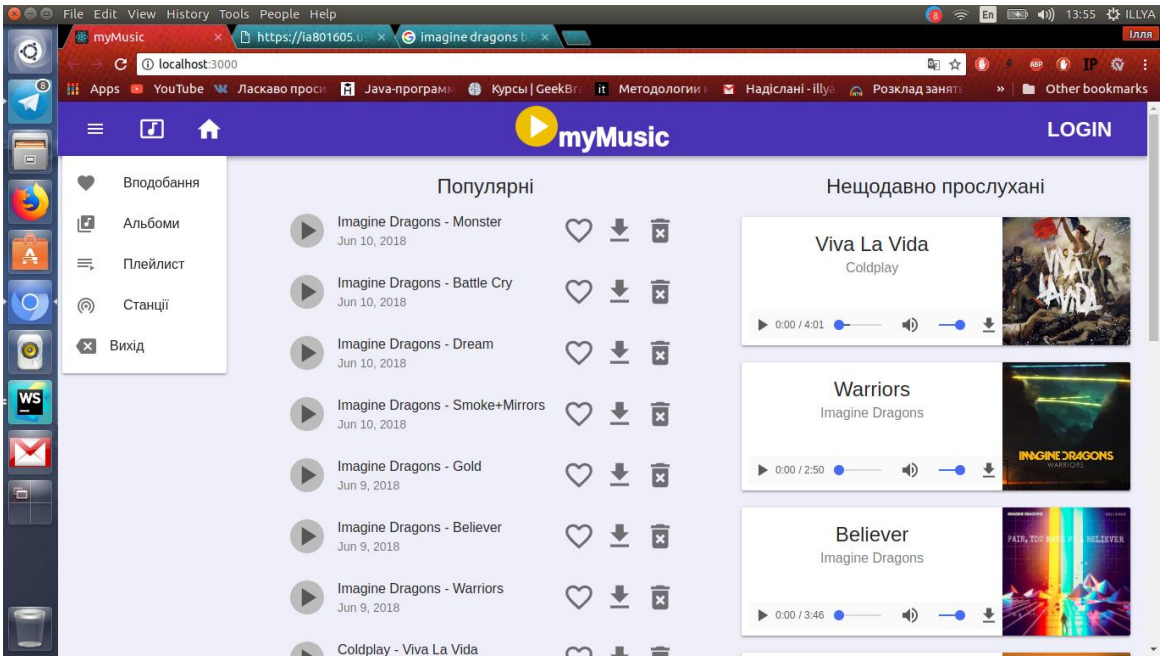


Рисунок 3.7 – Головна сторінка веб-сервісу

Тіло профільної сторінки користувача складається з таких компонентів: «profile», «profileHeader», «navbar», «profileShare».

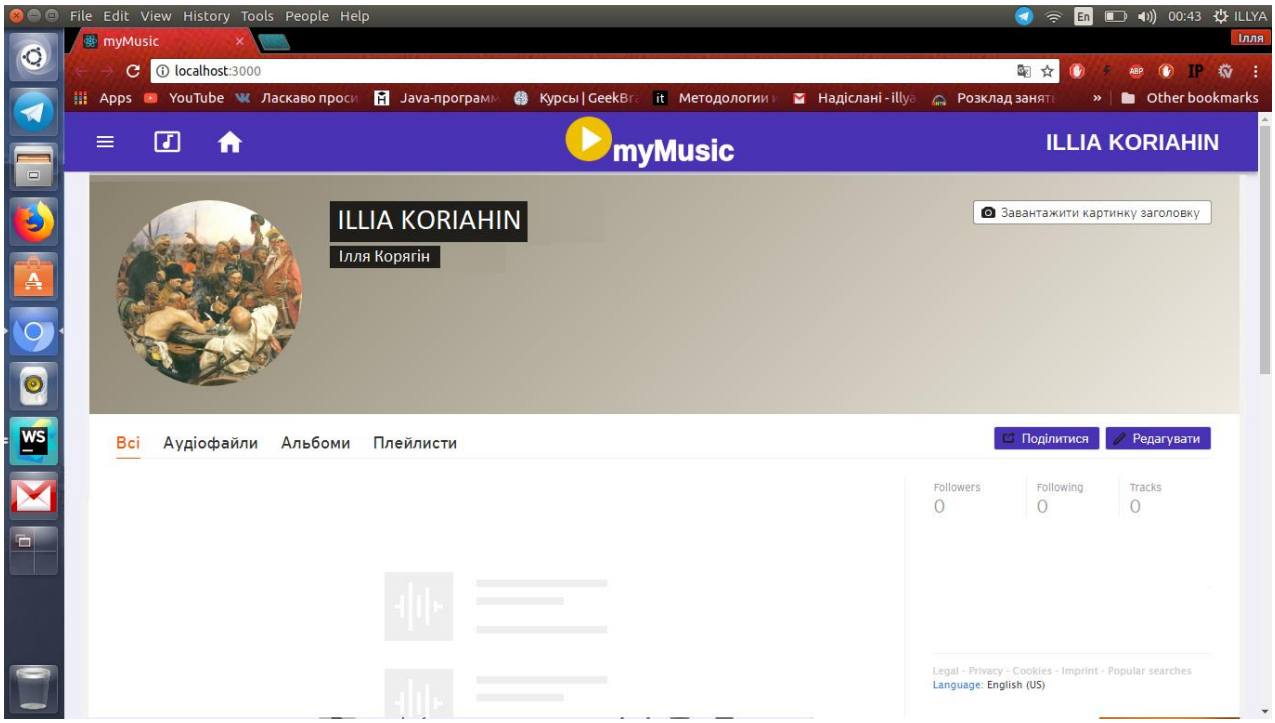


Рисунок 3.8 – Профіль користувача

Тіло сторінки завантаження складається з «downloader» компонента.

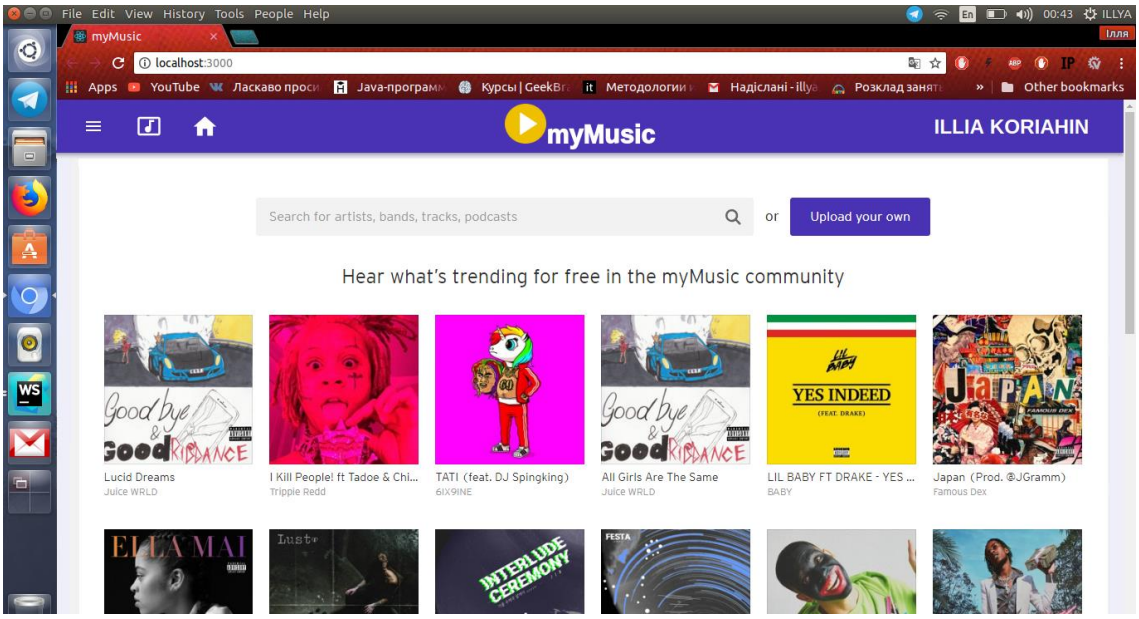


Рисунок 3.9 – Сторінка завантаження

Тіло головної сторінки складається з таких компонентів: «app», «navbar», «footer», «player».

Для реалізації методів реактивного виведення даних використовуються також інші модулі та компоненти, реалізація яких наведена у додатку Б.

3.6 Висновки

Таким чином, проведено аналіз мов програмування, зокрема, у табличній формі, а також середовищ розробки. Серед мов програмування – JavaScript, PHP, TypeScript – обрано мовою розробки – TypeScript, оскільки, підсумувавши коефіцієнти у кожній мові програмування, найбільше значення «3.8» у TypeScript. TypeScript надає всі необхідні засоби для швидкої реалізації додатку.

Також, проведено аналіз середовищ для розробки веб-сервісу – PhpStorm, WebStorm, Aptana Studio, Eclipse. В ході аналізу вирішено, що середовищем розробки буде WebStorm, оскільки воно найкраще підходить для розробки методів реактивного виведення даних. Розглянуто розробку методу селективного вибору.

Продемонстровано роботу візуальної частини після розробки методів реактивного виведення даних.

4 ТЕСТУВАННЯ МЕТОДІВ

4.1 Методи тестування

Тестування програмного забезпечення – це:

1. Процес дослідження ПЗ з метою отримання інформації про якість продукту;
2. Процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснений шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином;
3. Оцінка системи для того, щоб знайти відмінності між тим, якою система повинна бути і якою вона є.

У широкому сенсі, тестування – це одна з технік контролю якості (Quality Control), яка включає планування, складання тестів, безпосередньо виконання тестування і аналіз отриманих результатів[18].

Важливо розуміти, що тестування ПЗ включає в себе не тільки проведення тестів, але і багато інших дій, пов’язаних з процесом забезпечення якості:

1. Аналіз і планування.
2. Розробку тестових сценаріїв.
3. Оцінку критеріїв закінчення тестування.
4. Написання звітів.
5. Рецензування документації (в тому числі і вихідного коду).
6. Проведення статичного аналізу.

Розглянемо наступні методи тестування: «біла скринька» і «чорна скринька».

Метод тестування «біла скринька» виконується розробником (так як необхідно знати внутрішні принципи роботи програми) для перевірки внутрішньої структури програмного засобу. Об’єктом тестування є дані, отримані шляхом аналізу логіки програми. Перевіряється коректність побудови всіх елементів програми та правильність їхньої взаємодії один з одним. Зазвичай

аналізуються керуючі зв'язки елементів, рідше – інформаційні зв'язки. Тестування за принципом «білої скриньки» характеризується ступенем, в якому тести виконують або покривають логіку (вихідний текст) програми. Зазвичай тестування «білої скриньки» засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління[19].

Принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

1. Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.
2. Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.
3. Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.
4. При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

Метод тестування «чорна скринька» використовується, коли тестувальнику не потрібно знати внутрішні властивості програми. При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тести демонструють:

1. Як виконуються функції програми.
2. Як приймаються вихідні дані.
3. Як виробляються результати.
4. Як зберігається цілісність зовнішньої інформації.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми. Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

1. Некоректних чи відсутніх функцій;
2. Помилок інтерфейсу;
3. Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;
4. Помилок характеристик (необхідна ємність пам'яті і т. д.);
5. Помилок ініціалізації та завершення.

На основі розглянутих методів існує також тестування «сіра скринька». При роботі з даним методом тестувальник має доступ до коду програми, проте тестування проводить з точки зору кінцевого користувача. Суть даних методів не є складною, проте ефективність тестування за допомогою кожного з них вимагає хороших знань та навичок. В результаті, обрано метод тестування «чорна скринька», оскільки за даним методом тестуються лише вхідні/вихідні дані.

Оскільки розроблено веб-сервіс, доцільним є провести тестування графічного інтерфейсу користувача для забезпечення його відповідності до даної специфікації. Завданням тестування графічного інтерфейсу користувача є виявлення помилок наступного характеру:

1. Помилки у функціональності за допомогою інтерфейсу.
2. Необроблені виключення при взаємодії з інтерфейсом.
3. Втрата або перекручування даних, переданих через елементи інтерфейсу.

4. Помилки в інтерфейсі (невідповідність проектної документації, відсутність елементів інтерфейсу).

Важливим для тестування інтерфейсу користувача є оцінка його зручності. Зручність використання графічного інтерфейсу виявляє міру простоти доступу користувача до функцій системи, наданих через інтерфейс програми. Тестування зручності використання інтерфейсу не належить до класичного методу тестування програмних систем. Фахівець по тестуванню користувальницького інтерфейсу має поєднувати у собі знання в галузі як програмної інженерії, так і фізіології, психології та ергономіки[20].

На зручність використання користувальницького інтерфейсу впливають такі чинники:

1. Легкість навчання – чи швидко людина навчається використовувати систему.
2. Ефективність навчання – чи швидко людина працює після навчання.
3. Помилки – чи часто людина припускається помилок у роботі.
4. Загальна задоволеність – чи є загальне враження від співпраці з системою позитивним.

Усі ці фактори, незважаючи на неформальність, можуть бути виміряні. Для таких вимірів вибирається група типових користувачів. Під час їх роботи вимірюються показники роботи системи, і їм пропонується висловити враження від інтерфейсу за допомогою заповнення опитувальних аркушів[21].

4.2 Інструкція тестування методів реактивного виведення даних

Для методів реактивного виведення даних веб-сервісу «myMusic» проводиться тестування графічного інтерфейсу користувача з використанням методу «чорна скринька». Для перевірки правильності роботи додатку, протестуємо процес роботи загальних функцій, а також процес реактивного виведення даних у кожному з модулів. Розглянемо процес роботи веб-сервісу, його основних компонентів та функцій. Користувач відкриває головну сторінку

веб-сервісу (рис. 4.1), завантажуючи компонент App.js, який збирає в собі представлення усіх компонентів, задіяних на головній сторінці. Дана сторінка в такому випадку є гостьовою і має обмежений функціонал.

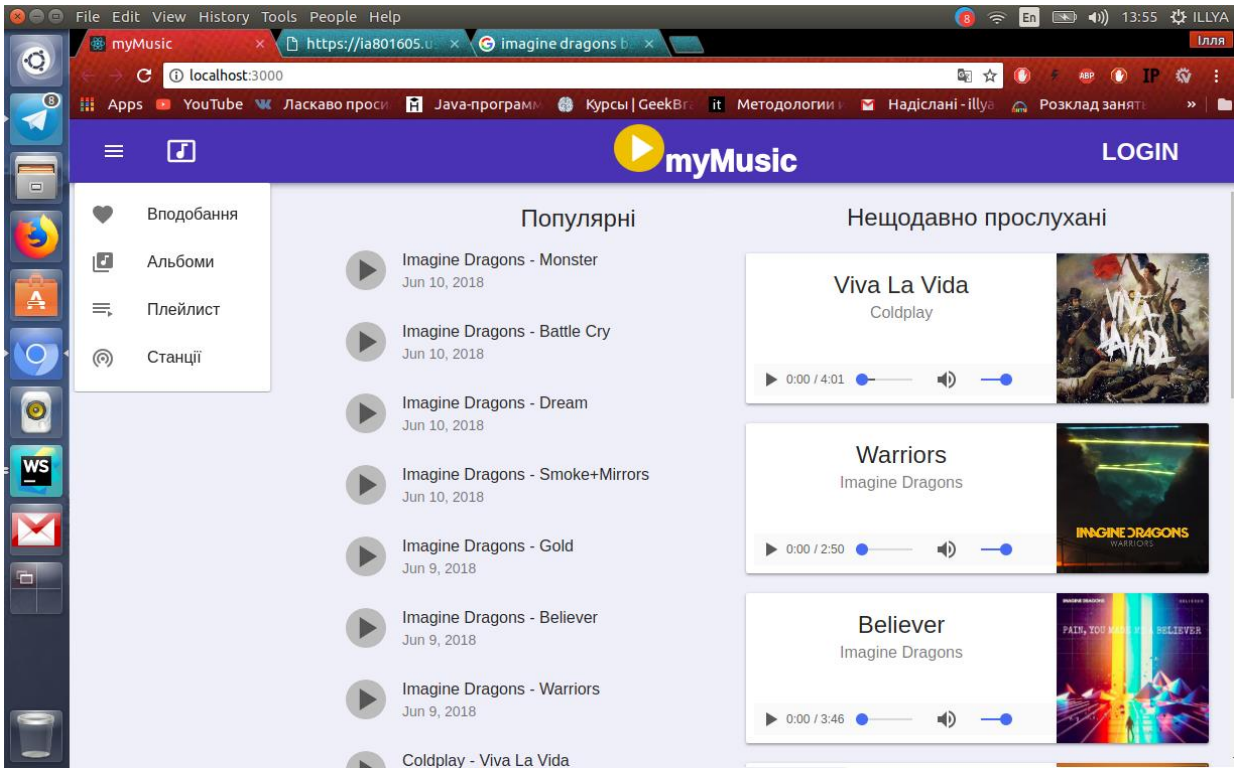


Рисунок 4.1 – Головна сторінка веб-сервісу

Для того, щоб отримати повний функціонал і відчути всі переваги «myMusic» слід авторизуватися натиснувши в правому верхньому куті кнопку «LOGIN», здійснивши перехід на сторінку авторизації (рис 4.2).

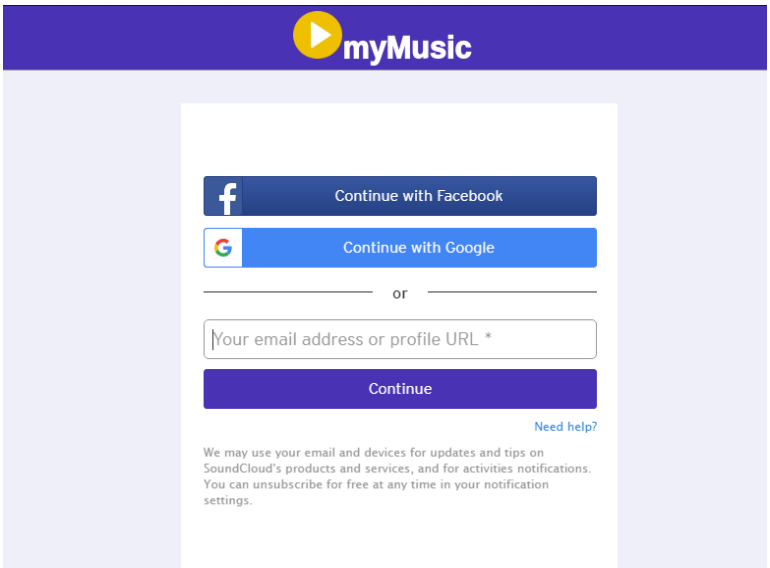


Рисунок 4.2 – Сторінка авторизації користувача

Після авторизації користувач повертається на головну сторінку, маючи повний доступ до всіх функцій системи, а саме: на заголовкі веб-сервісу з’являється домашня сторінка, де відображаються всі колекції користувача разом з нещодавно прослуханими аудіофайлами. Також стає доступною можливість видалення, скачування, а також завантаження і зберігання власних аудіофайлів тощо. Після авторизації головна сторінка виглядатиме наступним чином (рис. 4.3).

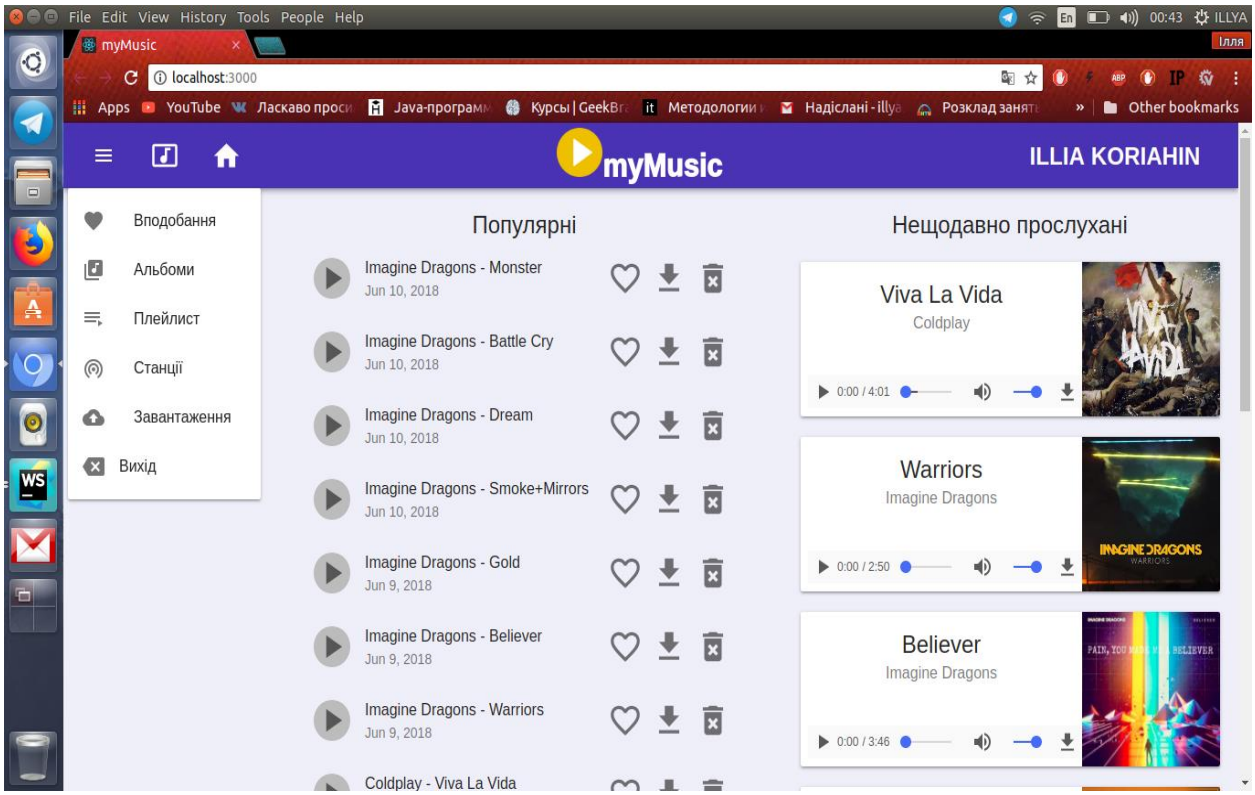


Рисунок 4.3 – Головна сторінка після авторизації

Розглянемо завантаження і зберігання аудіофайлів.

Для того, щоб завантажити власний аудіойфайл до веб-сервісу необхідно в компоненті Menu натиснути пункт меню «Завантаження», здійнивши перехід на сторінку завантаження і зберігання аудіофайлів.

Підчас переходу між сторінками веб-сервісу використовується сховище даних, тому швидкодія значно підвищена завдяки розробці методів реактивного виведення даних.

Сторінка завантаження і зберігання аудіофайлів представлена на рисунку 4.4.

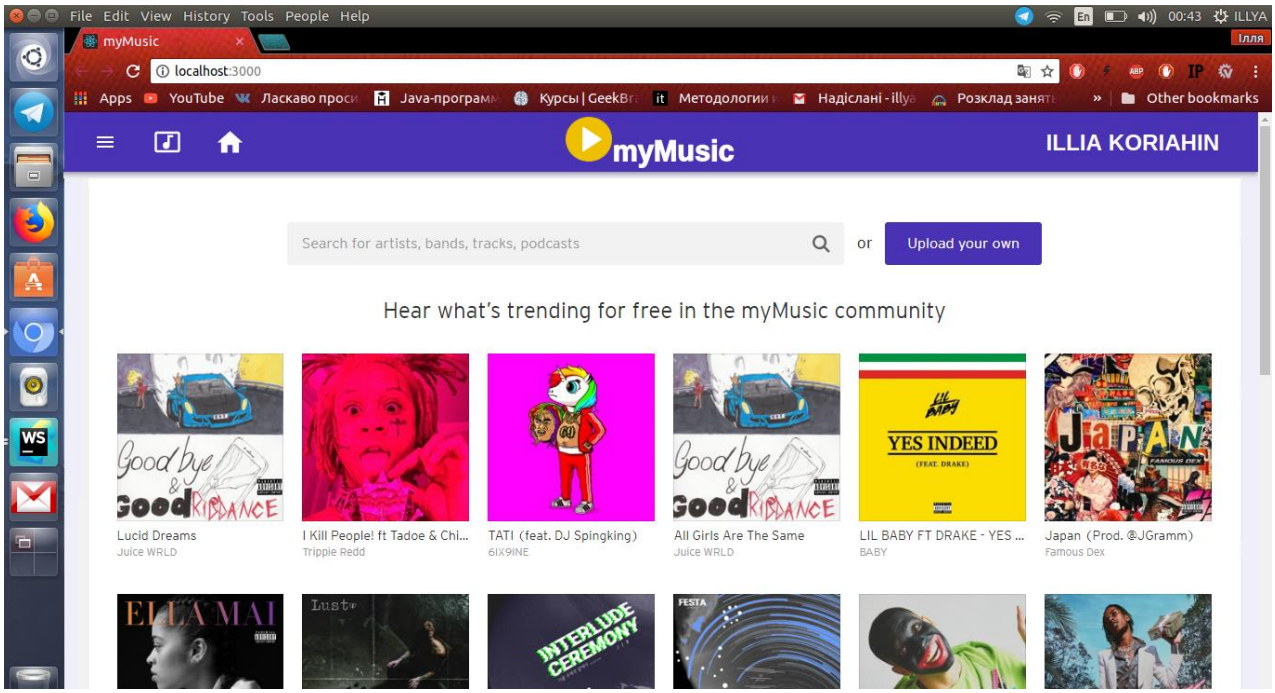


Рисунок 4.4 – Сторінка завантаження і зберігання аудіофайлів

Для перегляду і зміни персональних даних користувачу необхідно натиснути на кнопку, яка містить прізвище та ім’я, які користувач ввів під час входу в веб-сервіс. Профіль користувача зображено на рисунку 4.5.

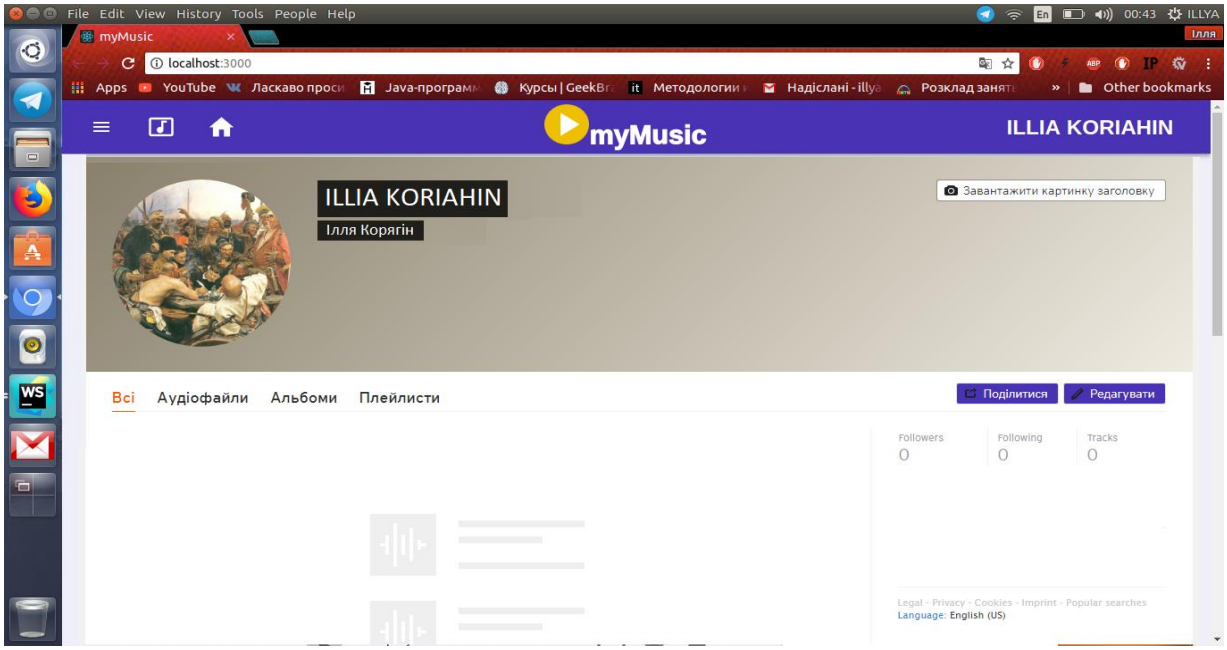


Рисунок 4.5 – Головна сторінка адміністратора системи

Після проведення тестування графічного інтерфейсу користувача з використанням методу «чорна скринька» виявлено повну відповідність між вхідними та вихідними даними, помилок при роботі програми не виявлено. Наявна система оброблення помилок у невідповідності введених даних. Тому методика тестування чорного ящика підтверджує нормальний режим роботи додатку.

Реактивне виведення даних користувачу наведено у додатку В на рисунках В.1 – В.6.

4.3 Висновки

В результаті, розглянуто методи тестування «чорної скриньки» та «білої скриньки». Виявлено як переваги, так і недоліки перелічених методів. Проте, оскільки для методів реактивного виведення даних веб-сервісу «myMusic» необхідно тестування вхідних / вихідних даних, обрано метод тестування «чорна скринька». Оскільки розробляються методи реактивного виведення даних, необхідним є тестування інтерфейсу користувача, щоб виявити та виправити помилки, що виникають під час роботи методів.

Проведено тестування всіх модулів додатку, а також допоміжних вікон (логування, реєстрація). В результаті виконання тестування помилок не було виявлено, цим самим підтверджено якісне функціонування методів.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами: Ракитянська Ганна Борисівна (к.т.н., доцент каф. ПЗ, ВНТУ), Майданюк Володимир Павлович (к.т.н., доцент каф. ПЗ, ВНТУ).

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями, що наведені в таблиці 5.1 [22].

Таблиця 5.1

Критерії оцінювання комерційного потенціалу розробки бальна оцінка

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

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні 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	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

Продовження таблиці 5.1

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 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років.Термін окупності 5-ти років	Термін реалізації ідеї менше 3-х років.Термін окупності від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів	Необхідно отримання великої кількості дозвільних документів, що вимагає значних коштів та часу	Процедура отримання дозвільних документів 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 комерційного потенціалу розробки наведено в таблиці 5.2.

Таблиця 5.2

Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Ракитянська	2. Майданюк
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	3	4
4	4	4
5	3	4
6	3	4
7	3	3
8	3	4
9	4	4
10	4	3

Продовження таблиці 5.2

11	3	4
12	3	4
Сума балів	$СБ_1 = 41$	$СБ_2 = 45$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43$	

Отже, з отриманих даних таблиці 5.2 видно, що нова розробка має високий рівень комерційного потенціалу. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 5.3.

Таблиця 5.3

Рівні комерційного потенціалу розробки

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

Рівень комерційного потенціалу розробки, становить 43 балів, що відповідає рівню «вище середнього».

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Проведемо прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько–технологічної роботи для розробки програмного забезпечення.

Виконаємо розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи, за такими статтями та формулами, приймаючи до уваги те, що для розробки інформаційної технології було залучено одного розробника програмного забезпечення.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 21 день;

t - число днів роботи розробника, t = 40 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 5.4.

Таблиця 5.4 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	7000	333,33	5	1666,65
Інженер-програміст	4500	214,28	40	8571,2
Всього:				10237,85

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 10237,85 = 1023,78 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \quad (5.2)$$

$$Н_{зп}= (10481,41+1048,14)\cdot \frac{36,3}{100} = 4185,22 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot\frac{Т}{12}, \quad (5.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 5.5

Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
Персональний комп'ютер	10000	25	3	625
Всього:				625

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.6 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	150	1	150
Пачка паперу	уп.	120	1	120
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				308

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п\ ; \quad (5.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=180 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0{,}8$).

$$В_е= 1{,}7 \cdot 0{,}6\ 180\cdot\ 0{,}8 = 146{,}88 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (10237{,}85 + 1023{,}78) = 11261{,}63 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 10237,85+1023,78+4087,97+562,5+146,88+308+11261,63= 27682,61 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{27682,61}{1} = 27682,61$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{27682,61}{0,9} = 30698,45 \text{ (грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності.

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (5.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

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

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

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

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

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 25 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 25 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 300 користувачів, протягом другого року – на 200 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 1500 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 200 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 25 \cdot 1500 + (200 + 25) \cdot 300 = 105000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 25 \cdot 1500 + (200 + 25) \cdot (300 + 200) = 150000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 25 \cdot 1500 + (200 + 25) \cdot (300 + 200 + 100) = 172500 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \qquad (5.10)$$

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

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3, 4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

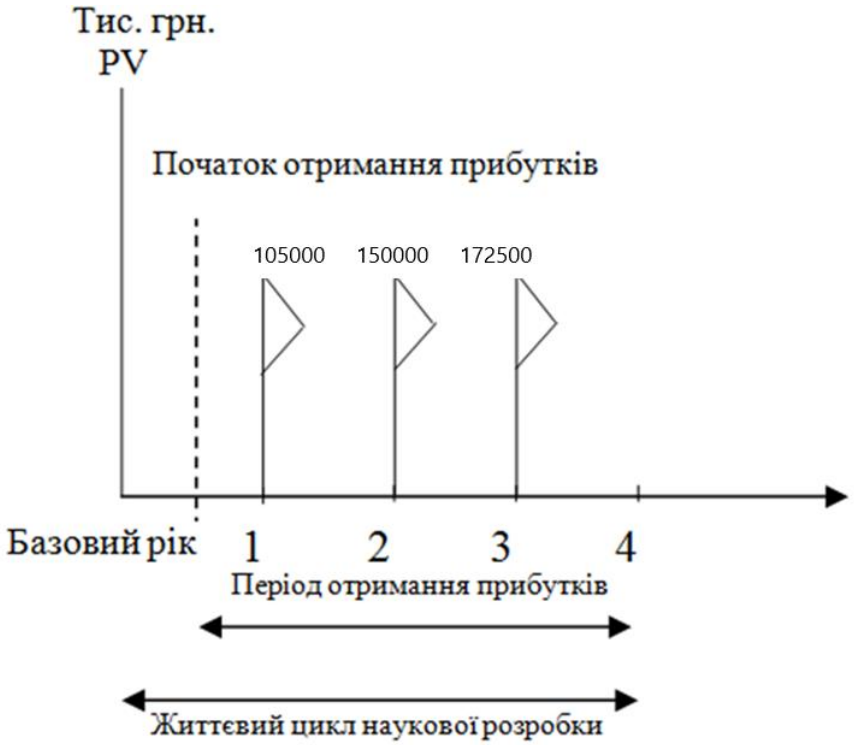


Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (5.11)$$

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

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{30698,45}{(1+0,1)^0} + \frac{105000}{(1+0,1)^2} + \frac{150000}{(1+0,1)^3} + \frac{172500}{(1+0,1)^4} = 347992,34 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 347992,34 - 30698,45 = 317293,89 \text{ грн.}$$

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

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (5.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{317293,89}{30698,45}} - 1 = 1,24 \text{ або } 124\ \%$$

Далі, розраховану величина $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в = 124\% > \tau_{мін} = 0{,}3 = 30\%$, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{1{,}24} = 0{,}80 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновки

В даному розділі було виконано оцінювання комерційного потенціалу розробки методів реактивного виведення даних.

Проведено технологічний аудит з залученням двох незалежних експертів. Визначено, що рівень комерційного потенціалу розробки є високим.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальні витрати на розробку складають 30698,45 грн. Про абсолютну ефективність вкладених інвестицій в сумі 317293,89 свідчить отримання прибутку інвестором від комерціалізації програмного продукту.

Щорічна ефективність вкладених у наукову розробку інвестицій складе 124%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 70%. Це може свідчити про зацікавленість інвесторів у фінансуванні нової розробки.

Термін окупності вкладених у реалізацію проекту інвестицій становить 0,8 року, що також свідчить про доцільність фінансування нової розробки.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було проаналізовано існуючі аналоги методів реактивного виведення даних, у тому числі методу проміжного збереження даних. Побудовано структурні схеми архітектури методів та структури глобального сховища даних. Було обрано мову програмування та інтегроване середовище розробки для створення додатку.

Описано програмну реалізацію методів реактивного виведення даних. Проаналізовано та обрано технічні засоби. Розроблено метод селективного вибору даних та продемонстровано розробку візуальної частини веб-сервісу. Було проаналізовано та обрано модель тестування методів.

Було розроблено методи реактивного виведення даних для веб-сервісу зберігання та відтворення аудіофайлів.

Методи розроблені засобами мови TypeScript в інтегрованому середовищі розробки JetBrains WebStorm 2017. В ході розробки було використано Angular, бібліотеки RxJS та NGXS, тому що дані технології найкраще реалізовують поставлені задачі.

В додатку було реалізовано наступні функціональні можливості:

- можливість проміжного зберігання даних;
- можливість потокового виведення даних;
- можливість селективного виведення даних;
- можливість автоматичного відслідковування змін;
- можливість реактивного оновлення даних;

Розроблена інструкція тестування програмного продукту. Було описано виняткові ситуації, що можуть виникнути під час роботи програмного додатку. Для тестування була обрана стратегія «чорної скриньки». Проведено тестування графічного інтерфейсу додатку з використанням методу тестування – «чорна скринька». Внаслідок перевірки роботи методів на відповідність очікуваного та фактичного результатів не виявлено жодних помилок.

Також, виконано розрахунок економічного ефекту від можливого впровадження розроблених методів і засобів, в ході якого підтверджено доцільність фінансування цієї наукової розробки.

Підсумовуючи усе вищесказане, можна зробити висновок, що задачі магістерської кваліфікаційної роботи було виконано у повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ReactiveX. URL: http://reactivex.io/intro.html.
2. Reactive programming. URL: https://en.wikipedia.org/ wiki/Reactive.
3. Panisuan Chasinga. Go Reactive Programming. : Packt Publishing , 2017. — 80 p.
4. Observable. URL: http://reactivex.io/documentation/observable.html.
5. NGXS. URL: https://www.ngxs.io/.
6. Порівняльний аналіз javascript фреймворків для розробки мережевого журналу. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-018/paper/view/5232.
7. Node.js. URL: https://uk.wikipedia.org/wiki/Node.js
8. Битва титанов: Angular 2 vs React. Что выбирают в 2017 году? URL: https://blog.ithillel.ua/articles/bitva-titanov-angular-2-vs-react.-chto-vybirayut-v-2017-godu
9. React. URL:http://thewebland.net/development/javascript/react/architecture/
10. Angular 2. URL: http://thewebland.net/development/javascript/angular2/-architecture
11. JavaScript. URL: https://uk.wikipedia.org/wiki/ JavaScript
12. TypeScript – клієнтська мова програмування, що робить сторінки сайту інтерактивними. URL: http://webdreamlab.com/tehnology/typescript.html
13. WebStorm. URL: https://uk.wikipedia.org/wiki/WebStorm
14. PhpStorm. URL: https://uk.wikipedia.org/wiki/ PhpStorm
15. Eclipse. URL: https://uk.wikipedia.org/wiki/ Eclipse
16. Alex Libby. Instant LESS CSS Preprocessor How-to.: Packt Publishing (англ.)русск., 2013. — 80 p.
17. Вільна енциклопедія. Інтерфейс користувача. URL: http://uk.wikipedia.org/-wiki/Інтерфейс_користувача
18. Вільна енциклопедія. Software testing. URL: https://en.wikipedia.org/wiki/-Software_testing

19. Степанченко И.В. Методы тестирования программного обеспечения: Учеб. Пособие – Волгоград: ВолгГТУ , 2006. – 74 с.

20. Калбертсон Р. Быстрое тестирование.: М.: «Вильямс», 2002. — 374 с.

21. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. СПб.: Питер, 2004. — 320 с.

22. Козловський В. О. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт. Вінниця: ВНТУ. 2012.

ДОДАТКИ

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

"____ "_____2019 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методів реактивного виведення даних для веб-сервісу зберігання та відтворення аудіофайлів»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к.т.н., доц. Г.Б. Ракитянська

"____"_____2019 р.

Виконав:

_____студент гр.1ПІ-18м І.С. Корягін

"____"_____2019 р.

Вінниця – 2019 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів реактивного виведення даних для веб-сервісу зберігання та відтворення аудіофайлів». Згідно отриманого завдання кінцевий програмний продукт може використовуватись офіційною організацією.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол № _____ від _____.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є підвищення продуктивності веб-сервісу зберігання та відтворення аудіофайлів за рахунок реактивного виведення даних.

Об’єктом дослідження є процес реактивного виведення даних.

Для досягнення поставленої мети в роботі вирішуються такі завдання:

- Розробка проміжного сховища даних Store.
- Надання можливості збереження багаторівневих структур даних.
- Підтримка різних типів даних.
- Підтримка потокового виведення даних.
- Надання можливості селективного вибору даних.
- Автоматичне відслідковування даних.
- Можливість обробки відповіді з серверу.
- Наявність початкового стану сховище даних.
- Швидкий доступ до сховище з будь-якого компоненту

4. Технічні вимоги

Вихідні дані до роботи: базові методи виведення даних, метод локального збереження даних, веб-сервіс зберігання та відтворення аудіофайлів, фреймворк Angular, мова програмування TypeScript, браузер Google Chrome, двухядерний процесор Intel з підтримкою технології Hyper-threading.

Вихідні методи для модифікації: бібліотека NGXS, бібліотека RxJS, фреймворк Angular, менеджер пакетів npm або yarn.

5. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

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. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз проблеми, обґрунтування актуальності розробки методів та постановка задачі	04.09.2019 – 29.09.2019
2	Розробка архітектури методів реактивного виведення даних	30.09.2019 – 26.10.2019
3	Розробка методів реактивного виведення даних	27.10.2019 – 3.11.2019
4	Тестування методів	4.11.2019 – 12.11.2019
5	Економічна частина	13.11.2019 – 17.11.2019

9. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

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. Лістинг програми

Auth.state.ts

```
@State<AuthStateModel>({
  name: 'auth',
  defaults: {
    initialized: false,
    loading: false,
    loaded: false,
    user: null,
    token: null,
  },
})
export class AuthState implements NgxsOnInit {
  constructor(
    private route: ActivatedRoute,
    private store: Store,
    private settings: Settings,
    private auth: AuthService,
    private scroll: ScrollService,
    private userService: UserService
  ) {}
  @Selector()
  static initialized(state: AuthStateModel): boolean {
    return state.initialized;
  }
  @Selector()
  static user(state: AuthStateModel): User {
    return state.user;
  }
  @Selector()
  static userName(state: AuthStateModel): string {
    const user = state.user;
    if (user.first_name || user.last_name) {
      return `${user.first_name} ${user.last_name}`;
    }
    if (user.username) {
```

```
      return user.username;
    }
    if (user.email) {
      return user.email;
    }
    return '';
  }
  @Selector()
  static isAuthenticated(state: AuthStateModel): boolean {
    return state.user ? state.user.is_authenticated : false;
  }
  @Selector()
  static isVerified(state: AuthStateModel): boolean {
    return state.user ? state.user.is_verified : false;
  }
  @Selector()
  static isLoading(state: AuthStateModel): boolean {
    return state.loading;
  }
  @Selector()
  static isLoaded(state: AuthStateModel): boolean {
    return state.loaded;
  }
  ngxsOnInit(ctx: StateContext<AuthStateModel>) {
    ctx.dispatch(new CheckSession());
    const routerState$ = this.store.select(state => state.router.state);
    routerState$.pipe(
      switchMap(
        routeState => this.store.select(AuthState.isAuthenticated).pipe(
          filter(isAuthenticated => isAuthenticated && has(routeState,
'data.loginRoute')),
        ),
      ),
    ).subscribe(() => ctx.dispatch(new
Navigate([this.settings.LOGIN_REDIRECT_URL])));
  }
  @Action(Init)
  resetPasswordInit(ctx: StateContext<AuthStateModel>) {
```

```
  ctx.patchState({
    loading: false,
    loaded: false,
  });
}

@Action(CheckSession)
checkSession(ctx: StateContext<AuthStateModel>) {
  return this.userService.getUser().pipe(
    take(1),
    tap((user: User) => {
      if (user.is_authenticated) {
        ctx.dispatch(new SessionSuccess(user));
      } else {
        ctx.dispatch(new SessionClosed(user));
      }
    }),
  );
}
@Action(SessionSuccess)
sessionSuccess(ctx: StateContext<AuthStateModel>, action: SessionSuccess) {
  ctx.patchState({
    initialized: true,
    loading: false,
    user: action.user,
  });
}
@Action(SessionClosed)
sessionClosed(ctx: StateContext<AuthStateModel>, action: SessionClosed) {
  const user = ctx.getState().user;
  const wasAuthenticated = user && user.is_authenticated;
  ctx.patchState({
    initialized: true,
    loading: false,
    user: action.user,
  });
  if (wasAuthenticated && !action.user.is_authenticated) {
    ctx.dispatch(new LoginRedirect());
```

```
    }
  }
  @Action(LoginWithEmailAndPassword)
  loginWithEmailAndPassword(ctx: StateContext<AuthStateModel>, action:
LoginWithEmailAndPassword) {
    ctx.patchState({loading: true, loaded: false});
    this.auth.signInWithEmailAndPassword(action.data)
      .subscribe(
        token => ctx.dispatch(new LoginSuccess(token)),
        err => ctx.dispatch(new LoginFail(err)),
      );
  }
  @Action(LoginSuccess)
  setUserStateOnSuccess(ctx: StateContext<AuthStateModel>, event: LoginSuccess)
{
    ctx.patchState({
      token: event.token,
    });
  }
  @Action(LoginFail)
  loginFail({patchState, dispatch}: StateContext<AuthStateModel>, {err}:
LoginFail) {
    patchState({loading: false, loaded: false});
    dispatch(new BackendError(err));
  }

  @Action(LoginRedirect)
  loginRedirect(ctx: StateContext<AuthStateModel>) {
    ctx.dispatch(new Navigate([this.settings.LOGIN_URL]));
  }

  @Action([LoginSuccess, SendFormSuccess])
  loginSuccess(ctx: StateContext<AuthStateModel>) {
    ctx.dispatch(new CheckSession());
    const redirect = this.route.snapshot.queryParams.redirect ||
this.settings.LOGIN_REDIRECT_URL;
    ctx.patchState({loading: false, loaded: true});
    ctx.dispatch(new Navigate([redirect]));
```

```
    this.scroll.toId('top');
  }
  @Action(Logout)
  logout(ctx: StateContext<AuthStateModel>) {
    return this.auth.logout().subscribe(
      () => ctx.dispatch(new LogoutSuccess()),
    );
  }
  @Action(LogoutSuccess)
  logoutSuccess(ctx: StateContext<AuthStateModel>) {
    const redirectUrl = this.settings.ACCOUNT_LOGOUT_REDIRECT_URL || '/';
    ctx.dispatch([
      ctx.dispatch(new CheckSession()),
      ctx.dispatch(new Navigate([redirectUrl])),
      ctx.dispatch(new DisablePreview())
    ]);
  }
  @Action(RegisterUser)
  registration(ctx: StateContext<AuthStateModel>, action: RegisterUser) {
    ctx.patchState({
      loading: true,
      loaded: false,
    });
    this.auth.registration(action.data).pipe(
      switchMap(token => this.userService.getUser()),
    ).subscribe(
      user => ctx.dispatch(new RegistrationSuccess(user)),
      err => ctx.dispatch(new RegistrationFail(err)),
    );
  }
  @Action(RegistrationSuccess)
  registrationSuccess(ctx: StateContext<AuthStateModel>, data:
RegistrationSuccess) {
    ctx.patchState({
      user: data.user,
      loading: false,
      loaded: true,
    });
```

```
    ctx.dispatch(new SendEmailConfirmation());

    ctx.dispatch(new Navigate([this.settings.LOGIN_REDIRECT_URL]));
    this.scroll.toId('top');
  }
  @Action(RegistrationFail)
  registrationFail(ctx: StateContext<AuthStateModel>, {err}: RegistrationFail) {
    ctx.patchState({
      loading: false,
      loaded: false,
    });
    ctx.dispatch(new BackendError(err));
  }

  @Action(ResetPassword)
  resetPassword(ctx: StateContext<AuthStateModel>, action) {
    ctx.patchState({
      loading: true,
      loaded: false,
    });
    this.auth.passwordReset(action.email).subscribe(
      data => ctx.dispatch(new ResetPasswordSuccess(data.detail)),
      err => ctx.dispatch(new BackendError(err)),
    );
  }

  @Action(ResetPasswordSuccess)
  resetPasswordSuccess(ctx: StateContext<AuthStateModel>, {detail}:
ResetPasswordSuccess) {
    ctx.patchState({
      loading: false,
      loaded: true,
    });
    ctx.dispatch(new SuccessMessage(detail));
    ctx.dispatch(new Navigate([this.settings.LOGIN_URL]));
  }
  @Action(ResetPasswordFail)
```

```
  resetPasswordFail(ctx: StateContext<AuthStateModel>, {err}: ResetPasswordFail)
{
    ctx.patchState({
      loading: false,
      loaded: false,
    });
    ctx.dispatch(new BackendError(err));
  }
  @Action(ResetPasswordConfirm)
  resetPasswordConfirm(ctx: StateContext<AuthStateModel>, action:
ResetPasswordConfirm) {
    ctx.patchState({
      loading: true,
      loaded: false,
    });
    this.auth.resetPasswordConfirm(action.passwords.new_password1,
      action.passwords.new_password2, action.uid, action.token).subscribe(
        data => ctx.dispatch(new ResetPasswordConfirmSuccess(data.detail)),
        err => ctx.dispatch(new ResetPasswordConfirmFail(err)),
      );
  }

  @Action(ResetPasswordConfirmSuccess)
  resetPasswordConfirmSuccess(ctx: StateContext<AuthStateModel>, {detail}:
ResetPasswordConfirmSuccess) {
    ctx.patchState({
      loading: false,
      loaded: true,
    });
    ctx.dispatch([
      new SuccessMessage(detail),
      new Navigate([this.settings.LOGIN_URL])
    ]);
  }

  @Action(ResetPasswordConfirmFail)
  resetPasswordConfirmFail(ctx: StateContext<AuthStateModel>, {err}:
ResetPasswordFail) {
```

```
    ctx.patchState({
      loading: false,
      loaded: false,
    });

    let hiddenError;
    const hiddenFields = ['uid', 'token'];
    for (const field of hiddenFields) {
      const errorLine = get(err, `error.${field}.0`, null);
      if (errorLine) {
        hiddenError = `${field}: ${errorLine}`;
        break;
      }
    }

    if (hiddenError) {
      ctx.dispatch([new BackendError(err), new ErrorMessage(hiddenError)]);
    } else {
      ctx.dispatch(new BackendError(err));
    }
  }

  @Action(SendEmailConfirmation)
  sendEmailConfirmation(ctx: StateContext<AuthStateModel>) {
    ctx.patchState({
      loading: true,
      loaded: false,
    });
    this.auth.sendEmailConfirmation().subscribe(
      resp => ctx.dispatch(new SendEmailConfirmationSuccess(resp.message)),
      err => ctx.dispatch(new SendEmailConfirmationFail(err)),
    );
  }

  @Action(SendEmailConfirmationSuccess)
  sendEmailConfirmationSuccess(ctx: StateContext<AuthStateModel>, {message}:
SendEmailConfirmationSuccess) {
    ctx.patchState({
```

```
      loading: false,
      loaded: true,
    });
    ctx.dispatch(new SuccessMessage(message));
  }

  @Action(SendEmailConfirmationFail)
  sendEmailConfirmationFail(ctx: StateContext<AuthStateModel>, {err}:
SendEmailConfirmationFail) {
    ctx.patchState({
      loading: false,
      loaded: false,
    });
    ctx.dispatch(new BackendError(err));
  }
  @Action(ConfirmEmail)
  confirmEmail(ctx: StateContext<AuthStateModel>, action: ConfirmEmail) {
    ctx.patchState({
      loading: true,
      loaded: false,
    });
    this.auth.confirmEmail(action.key).subscribe(
        data => ctx.dispatch(new ConfirmEmailSuccess(data)),
        err => ctx.dispatch(new BackendError(err)),
      );
  }
  @Action(ConfirmEmailSuccess)
  confirmEmailSuccess(ctx: StateContext<AuthStateModel>, {data}:
ConfirmEmailSuccess) {
    const user = ctx.getState().user;
    ctx.patchState({
      loading: false,
      loaded: true,
      user: {...user, is_verified: true},
    });
    const isAuthenticated = user.is_authenticated;
    let redirectUrl = null;
    if (isAuthenticated) {
```

```
      redirectUrl =
this.settings.ACCOUNT_EMAIL_CONFIRMATION_AUTHENTICATED_REDIRECT_URL
        || this.settings.LOGIN_REDIRECT_URL;
    } else {
      redirectUrl =
this.settings.ACCOUNT_EMAIL_CONFIRMATION_ANONYMOUS_REDIRECT_URL ||
this.settings.LOGIN_URL;
    }
    ctx.dispatch(new Navigate([redirectUrl]));

    const dataKey = 'detail';
    const message = data[dataKey] === 'ok' ? 'Your email was successfully
verified' : data[dataKey];
    timer(1000).subscribe(() => ctx.dispatch(new SuccessMessage(message)));
  }

  @Action(ConfirmEmailFail)
  confirmEmailFail(ctx: StateContext<AuthStateModel>, {err}: ConfirmEmailFail) {
    ctx.patchState({
      loading: false,
      loaded: false,
    });
    ctx.dispatch(new BackendError(err));
  }
  @Action([Loading])
  loading(ctx: StateContext<AuthStateModel>) {
    ctx.patchState({loading: true});
  }
  @Action([Loaded])
  loaded(ctx: StateContext<AuthStateModel>) {
    ctx.patchState({loading: false});
  }

  @Action(BackendError)
  backendError(ctx: StateContext<AuthStateModel>) {
    ctx.patchState({loading: false});
  }
```

```
}
```

Category.state.ts

```
import {Category, CategoryData} from '../models/category.model';
import { State, Action, StateContext, Selector, NgxsOnInit, Store } from
'@ngxs/store';
import { SetCurrentCategory, RemoveCurrentCategory, LoadCategory,
LoadCategorySuccess } from './category.action';
import { LoadProductsByIdCategory, AfterRemoveCategory, RemoveSearchingParam }
from './product.action';
import { BackendError } from 'src/app/messages/messages.actions';
import { CategoryService } from '../services/category.service';
export class CategoryStateModel {
  categories: Category[];
  categoryDict: {
    [id: number]: CategoryData
  };
  breadscrumbs: {
    [id: number]: number[],
  };
  currentCategory: number;
}
@State<CategoryStateModel>({
  name: 'category',
  defaults: {
    categories: null,
    categoryDict: {},
    breadscrumbs: {},
    currentCategory: null,
  },
})
export class CategoryState implements NgxsOnInit {
  constructor(
    private store: Store,
    private categoryService: CategoryService,
  ) { }

  @Selector()
  static getCurrentCategory(state: CategoryStateModel) {
```

```
    return state.currentCategory;
  }
  @Selector()
  static getCategories(state: CategoryStateModel) {
    return state.categories;
  }
  @Selector()
  static getPathDataById(state: CategoryStateModel) {
    const currentPath = state.breadscrumbs[state.currentCategory];
    return currentPath.map(p => ({ ...state.categoryDict[p], id : p }));
  }

  ngxsOnInit(ctx: StateContext<CategoryStateModel>) {

  }
  @Action(LoadCategory)
  onLoadCategory(ctx: StateContext<CategoryStateModel>) {
    if (!ctx.getState().categories) {
      this.categoryService.getCategoriesList().subscribe(
        data => ctx.dispatch(new LoadCategorySuccess(data)),
        err => ctx.dispatch(new BackendError(err)),
      );

    }

  }
  @Action(LoadCategorySuccess)
  onLoadCategorySuccess(ctx: StateContext<CategoryStateModel>, { data }:
LoadCategorySuccess) {

    const catDict = {};
    const catBreadcrum = {};
    function findPathCat(inputData, path = []) {
      inputData.forEach(category => {
        catDict[category.id] = category.data;
        if (!category.path) {
          category.path = [category.id];
        }
```

```
        if (path) {
          category.path = path.concat(category.path);
          category[category.id] = category.path;
          catBreadcrum[category.id] = category.path;
        }
        if (category.children) {
          findPathCat(category.children, category.path);
        }
      });
    }
    findPathCat(data);
    ctx.patchState({
      categories: data,
      categoryDict: catDict,
      breadscrumbs: catBreadcrum
    });
  }
  @Action(SetCurrentCategory)
  onSetCurrentCategory(ctx: StateContext<CategoryStateModel>, { id }:
SetCurrentCategory) {
    if (ctx.getState().currentCategory !== id) {
    ctx.patchState({
      currentCategory: id,
    });
    ctx.dispatch(new RemoveSearchingParam());
    ctx.dispatch(new LoadProductsByIdCategory(id));
   }
  }
  @Action(RemoveCurrentCategory)
  onRemoveCurrentCategory(ctx: StateContext<CategoryStateModel>) {
    if (ctx.getState().currentCategory) {
      ctx.patchState({
        currentCategory:  null,
      });
      this.store.dispatch(new AfterRemoveCategory());
    } else {
    this.store.dispatch(new RemoveSearchingParam());
    }
```

```
  }
}
```

Category.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Category as Categories, Category } from '../models/category.model';

@Injectable({
  providedIn: 'root'
})
export class CategoryService {
  constructor(private http: HttpClient) { }
  getCategoriesList(): Observable<Category[]> {
    return this.http.get<Category[]>('/api/audio/category/');
  }
}
```

Audio.state.ts

```
export interface AudioStateModel {
  dict: {[dict: string]: Product};
  items: {[key: string]: Item};
  selectedPage: number;
  selectedCategory: number;
  selectedSearch: string;
  loadedPages?: { [key: number]: boolean };
  currentItemKey?: string;
  currentElement?: number;
  length?: number;
  imgStatus?: number;
  recentlyViewed?: string[];
}
@State<AudioStateModel>({
  name: 'product',
  defaults: {
```

```
    dict: {},
    items: {},
    selectedPage: null,
    selectedCategory: null,
    selectedSearch: null,
    loadedPages: {},
    currentItemKey: null,
    currentElement: null,
    length: 50,
    imgStatus: NOT_DOWLOADED,
    recentlyViewed: [],
  }
})
……
@Selector([CategoryState])
  static getAudioList(ctx: AudioStateModel, categoryState: CategoryStateModel) {
      const page = ctx.selectedPage;
      const items = ctx.items;
      const categoryId = categoryState.currentCategory;
      const search = ctx.selectedSearch;
      const keysList = [];
      for (let index = 0; index <= page; index++) {
        const element = AudioState.itemKey(categoryId, index, search);
        if (items[element]) {
          keysList.push(element);
        }
      }
      const allListsIds = keysList.reduce((prev, k) =>
prev.concat(items[k].list), []);
      if (allListsIds.length) {
      ctx.length = allListsIds.length;
      }
      const results = allListsIds.reduce((previous, current) => {
        previous.push(ctx.dict[current]);
```

```
        return previous;
      }, []);
      return  results;
  }
…..
@Action(LoadCatalog)
  load(ctx: StateContext<AudioStateModel>, { page }: LoadCatalog) {
    const stateContext = ctx.getState();
    const key = AudioState.itemKey(null, page);
    const items = ctx.getState().items;
    const isKey = hasIn(items, key);
    if (!isKey) {
      this.productService.getProductList(page).subscribe(
        data => ctx.dispatch(new LoadCatalogSuccess(data, page, key)),
        err => ctx.dispatch(new LoadCatalogFail(err)),
      );
    } else {
        ctx.patchState({
          selectedPage: page,
          currentItemKey: key,
          loadedPages: {
            ...stateContext.loadedPages,
            [page]: true,
          }
        });
    }
  }
  @Action(Searching)
  searching(ctx: StateContext<AudioStateModel>, { search }: Searching) {
    const searchText = search.toLowerCase();
    const stateContext = ctx.getState();
    const key = AudioState.itemKey(null, 1, searchText);
    const items = ctx.getState().items;
    const isKey = hasIn(items, key);
```

```
    if (!isKey) {
      this.productService.getSearchProducts(searchText).subscribe(
        data => ctx.dispatch(new SearchingSuccess(data, searchText, key)),
        err => ctx.dispatch(new LoadCatalogFail(err)),
      );
    } else {
        ctx.patchState({
          selectedPage: 1,
          selectedSearch: searchText,
          currentItemKey: key,
          loadedPages: {
            ...stateContext.loadedPages,
            [1]: true,
          }
        });
        this.store.dispatch(
          new Navigate(['/catalogue' ], [`${searchText}`])
        );
    }
  }
  @Action(SearchingSuccess)
  SearchingSuccess(ctx: StateContext<AudioStateModel>, { searchProducts, search,
key }: SearchingSuccess) {
    this.store.dispatch(
      new Navigate(['/catalogue' ], [`${search}`])
    );
    const stateContext = ctx.getState();
    const products = searchProducts.results;
    const newList = [];
    const result = {};

    forOwn(products, (product) => {
      const productItem = product as Product;
      result[productItem.id] = productItem;
```

```
      newList.push(productItem.id);
    });

    ctx.patchState({
      loadedPages: {
        ...stateContext.loadedPages,
        [1]: true,
      },
      length: newList.length,
      currentItemKey: key,
      selectedPage: 1,
      selectedSearch: search,
      dict: {
        ...stateContext.dict,
        ...result
      },
      items: {
        ...stateContext.items,
        [key]: {
          ts: 12212,
          loading: false,
          loaded: true,
          next:  searchProducts.next,
          count: searchProducts.count,
          list: newList
          }
      },
    });
  }
  @Action(LoadCatalogSuccess)
  loadSuccess(ctx: StateContext<AudioStateModel>, { productData, page, key }:
LoadCatalogSuccess) {
    const stateContext = ctx.getState();
    const products = productData.results;
```

```
  const newList = [];
  const result = {};
  forOwn(products, (product) => {
    const productItem = product as Product;
    result[productItem.id] = productItem;
    newList.push(productItem.id);
  });
  ctx.patchState({
    loadedPages: {
      ...stateContext.loadedPages,
      [page]: true,
    },
    currentItemKey: key,
    selectedPage: page,
    dict: {
      ...stateContext.dict,
      ...result
    },
    items: {
      ...stateContext.items,
      [key]: {
        ts: 12212,
        loading: false,
        loaded: true,
        next:  productData.next,
        count: productData.count,
        list: newList
        }
    },
  });
}
```

Додаток В. Реактивне виведення даних користувачу

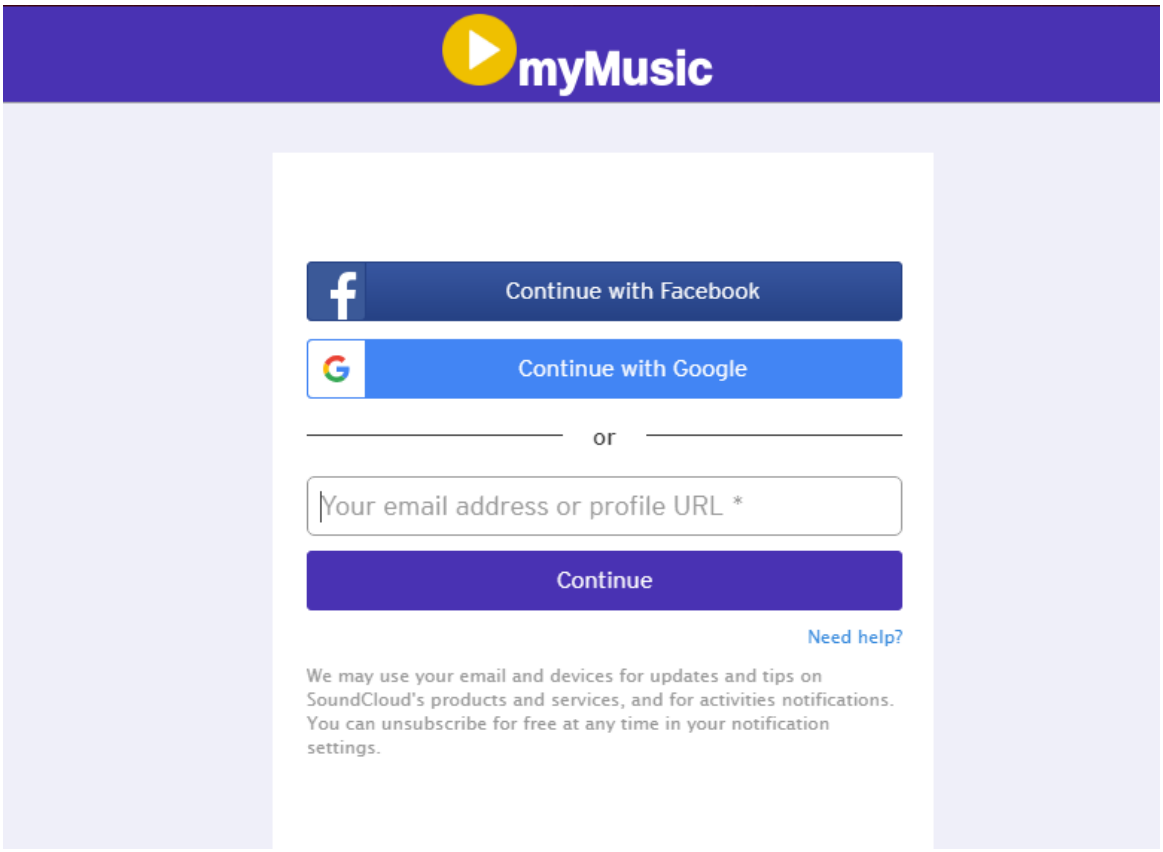


Рисунок В.1 – Сторінка авторизації

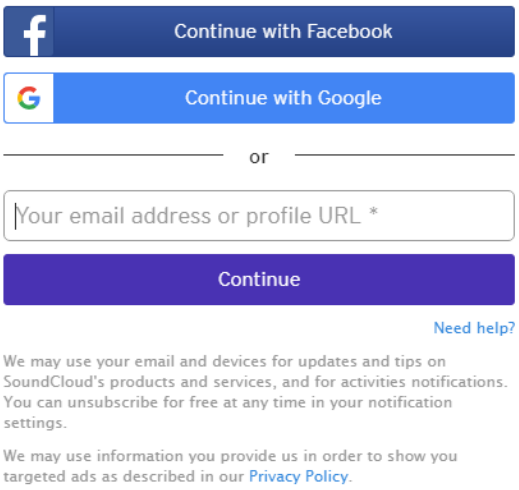


Рисунок В.2 – Сторінка реєстрації

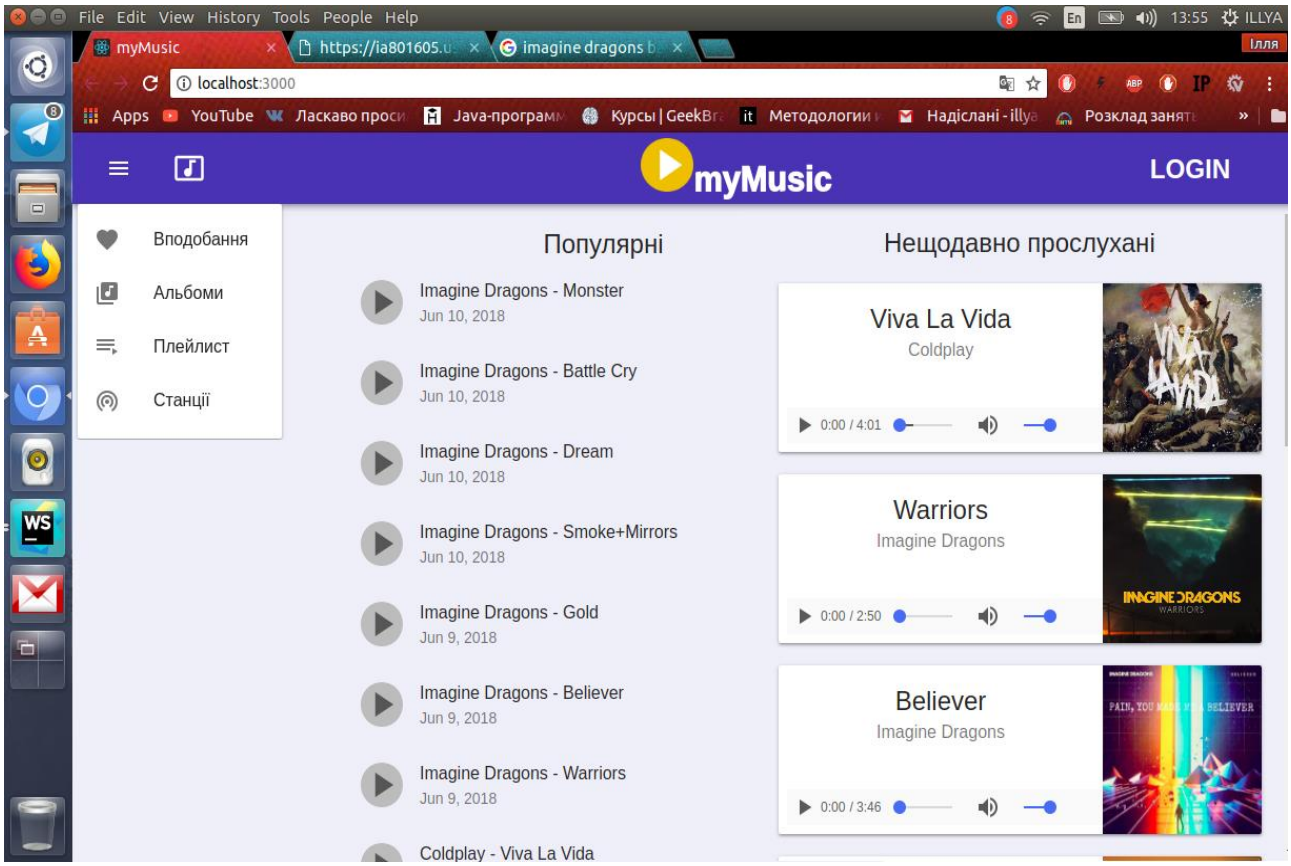


Рисунок В.3 – Головна сторінка веб-сервісу до авторизації

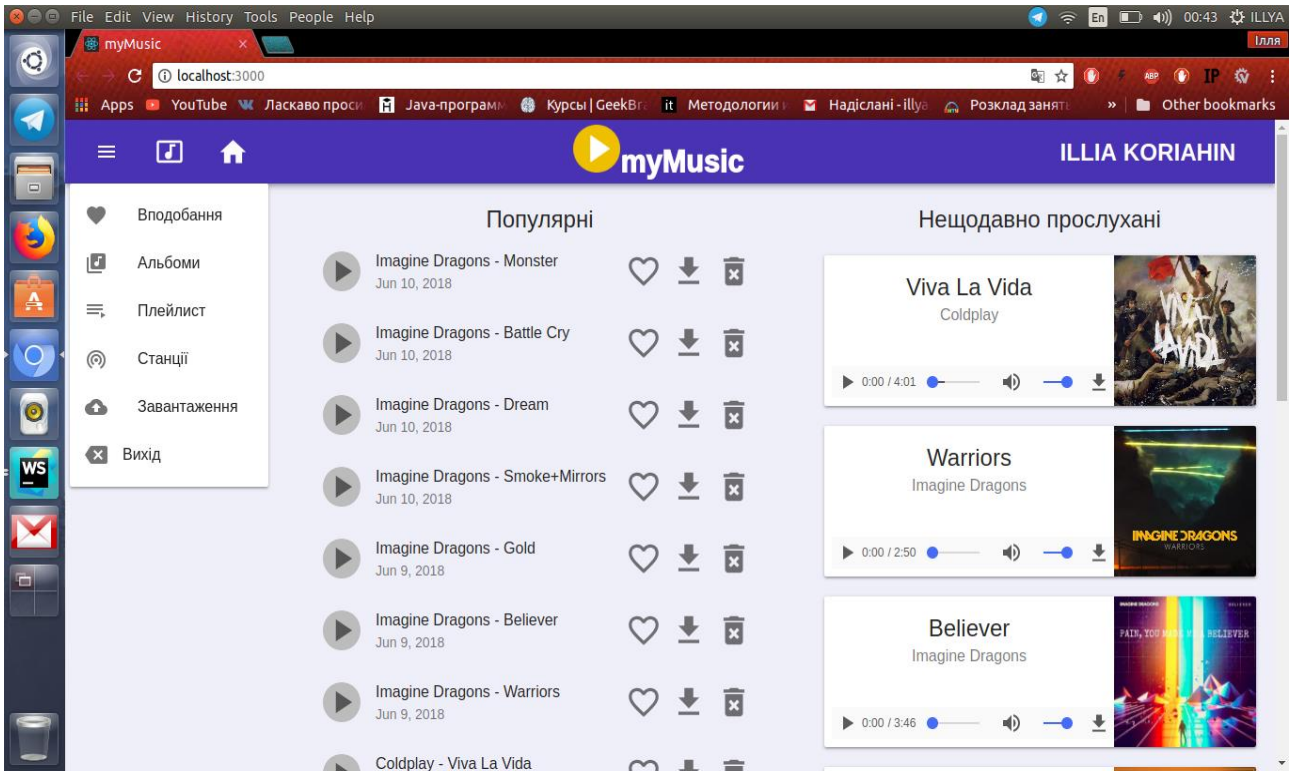


Рисунок В.4 – Головна сторінка веб-сервісу після авторизації

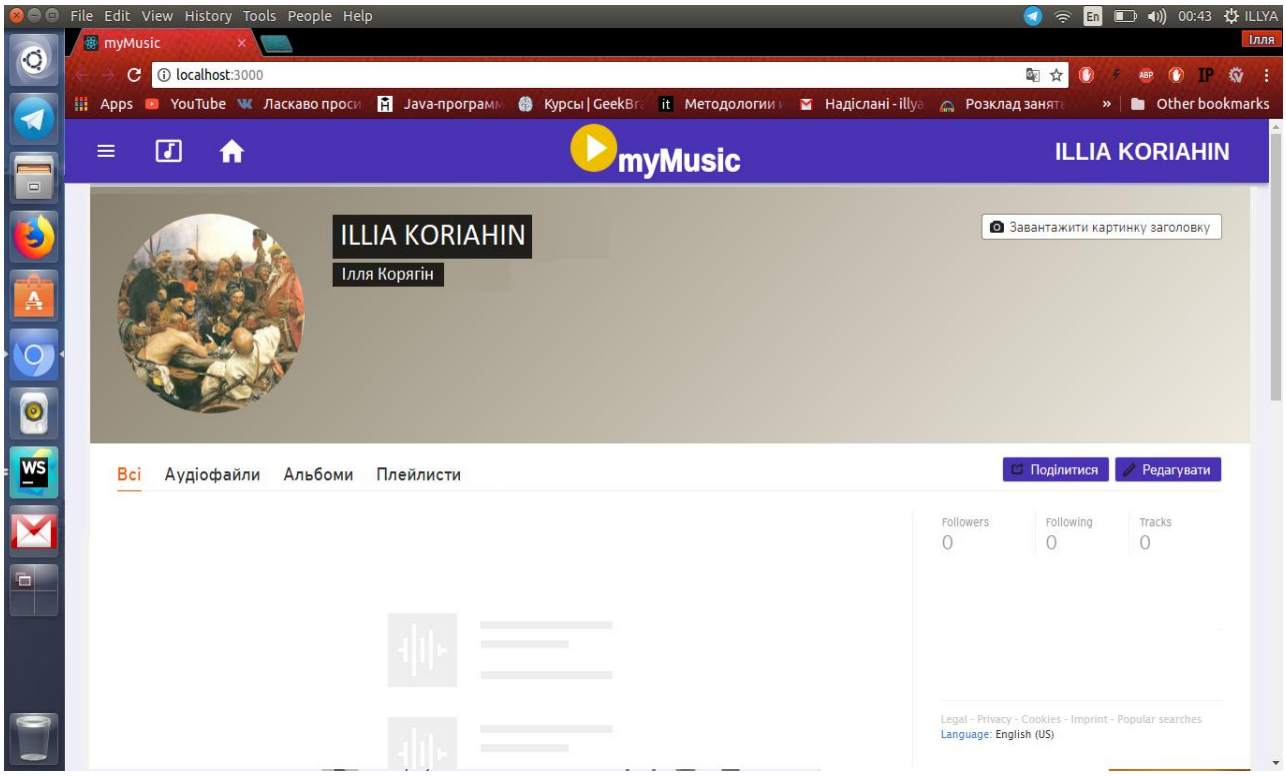


Рисунок В.5 – Профіль користувача

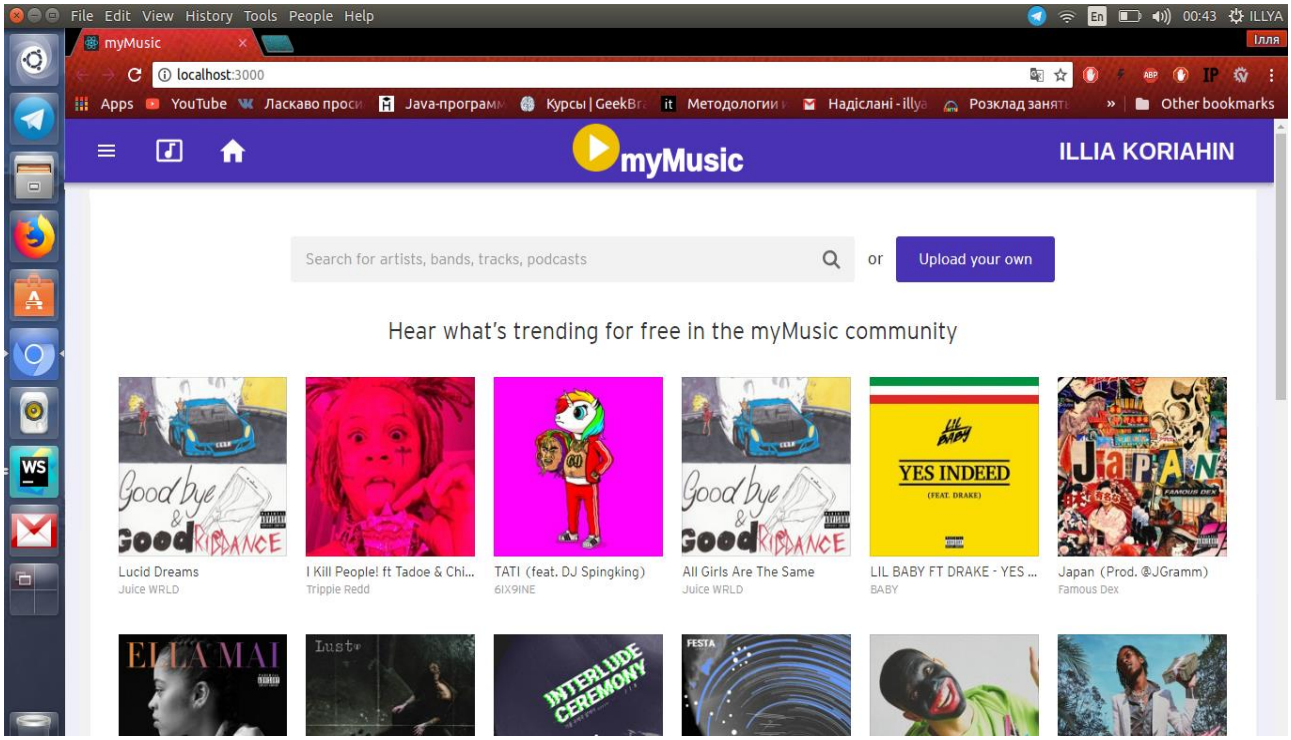


Рисунок В.6 – Сторінка завантаження і зберігання аудіофайлів

Додаток Г. Ілюстративний матеріал

Ілюстративний матеріал до захисту магістерської кваліфікаційної роботи

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

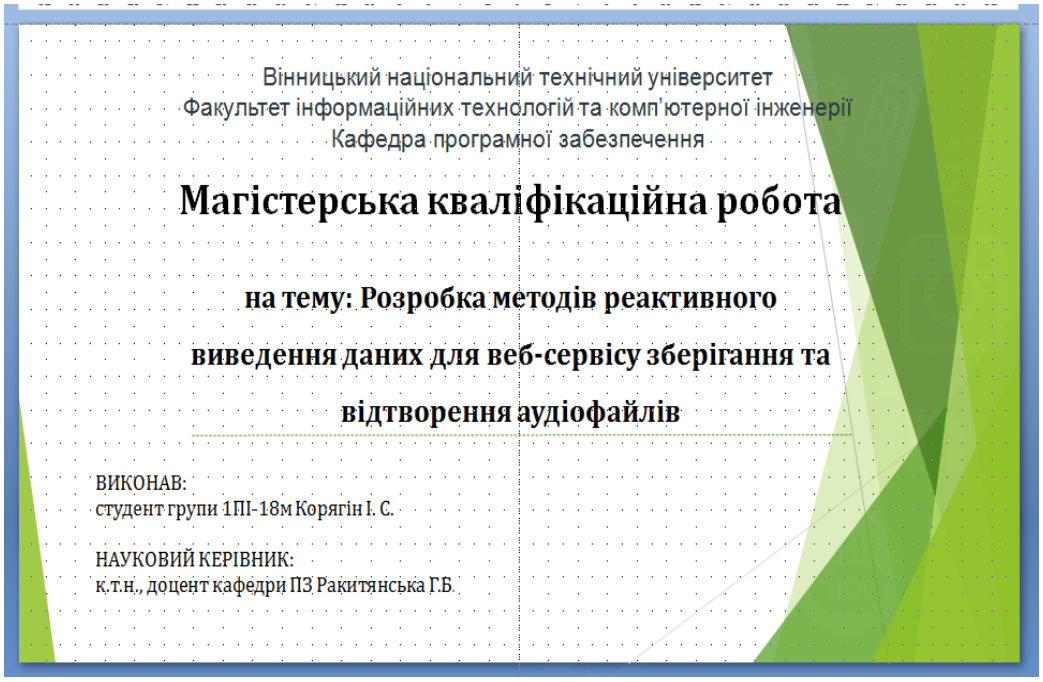
Науковий керівник, к. т. н., доцент кафедри ПЗ _____ Г. Б. Ракитянська

Рецензент, к. т. н., доцент кафедри КН _____ Я. В. Іванчук

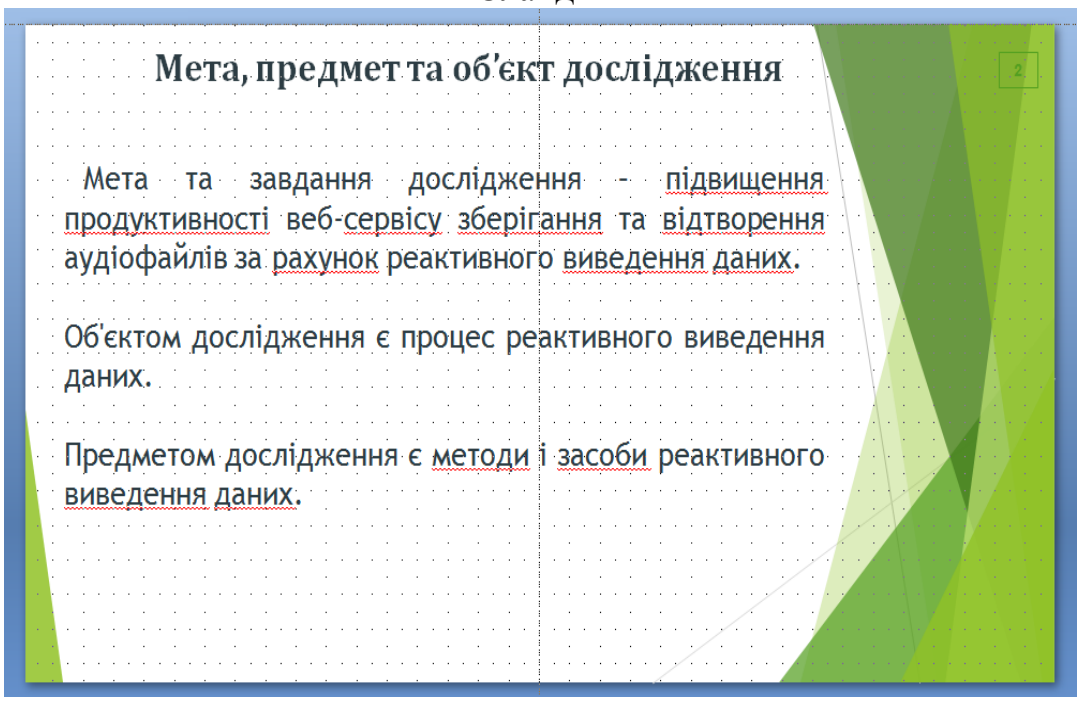
Нормоконтроль, к. т. н., доцент кафедри ПЗ _____ Г. Б. Ракитянська

Виконавець, студент гр.1ПІ-18м _____ І. С. Корягін

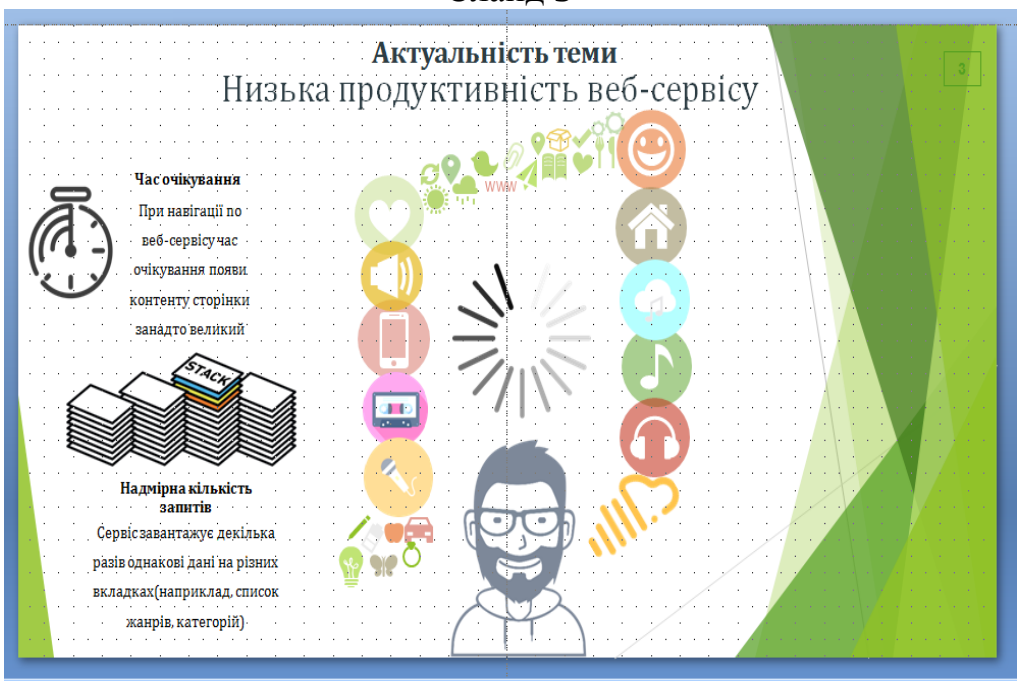
Слайд 1



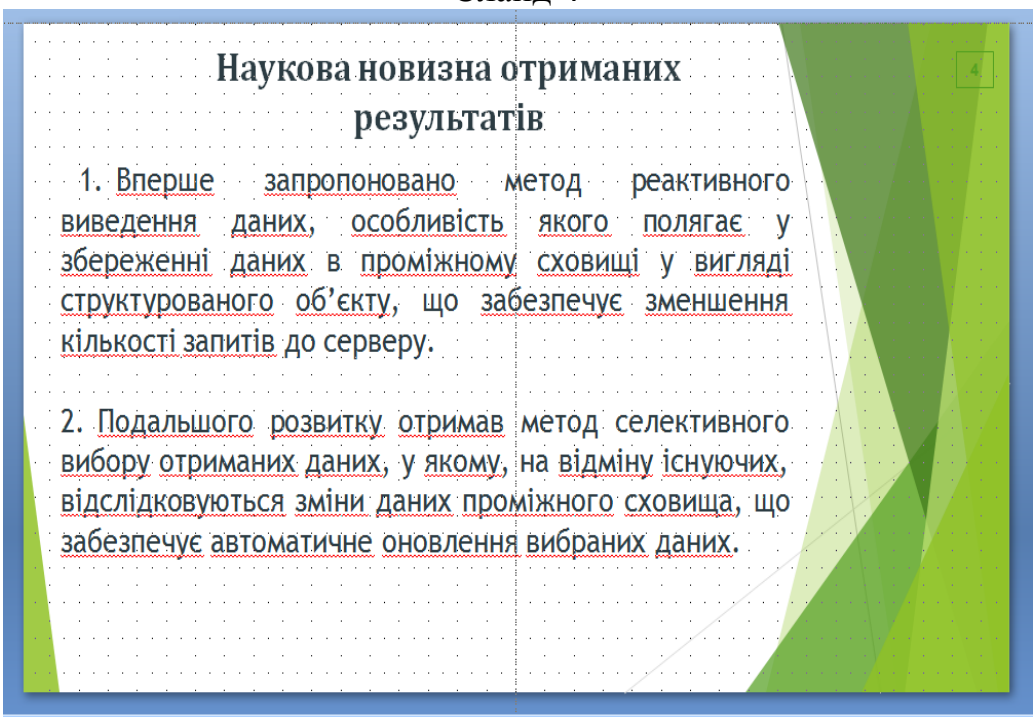
Слайд 2



Слайд 3

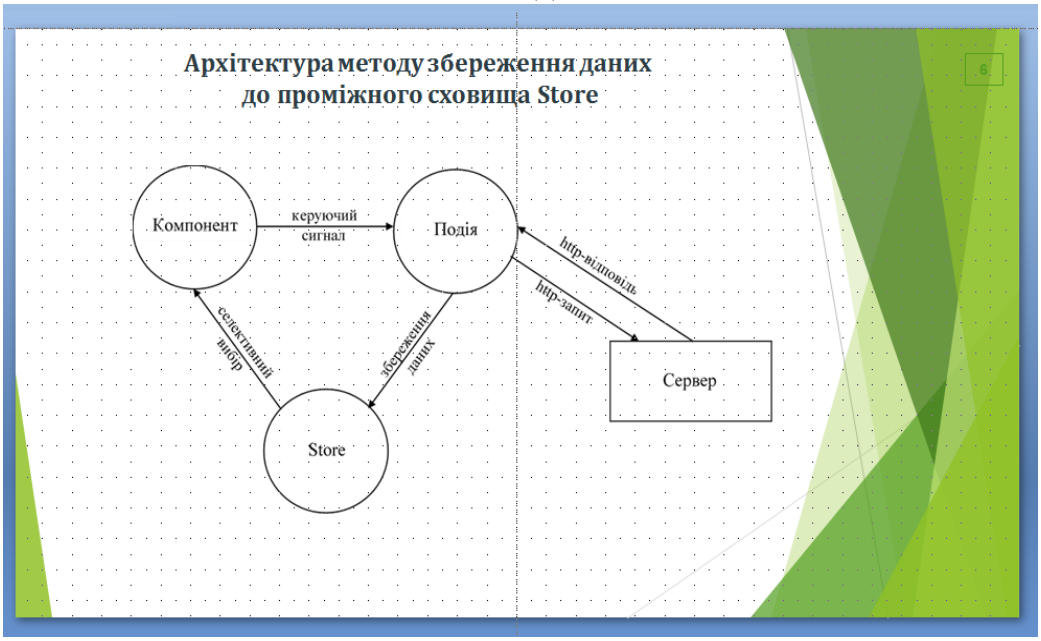


Слайд 4



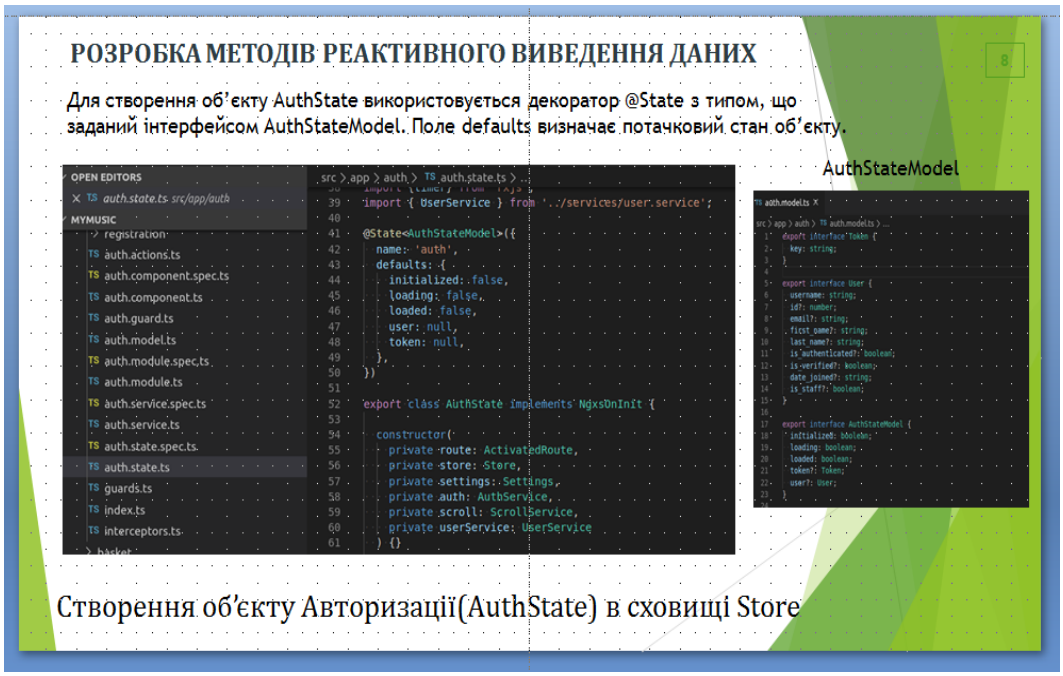
Слайд 5

Слайд 6

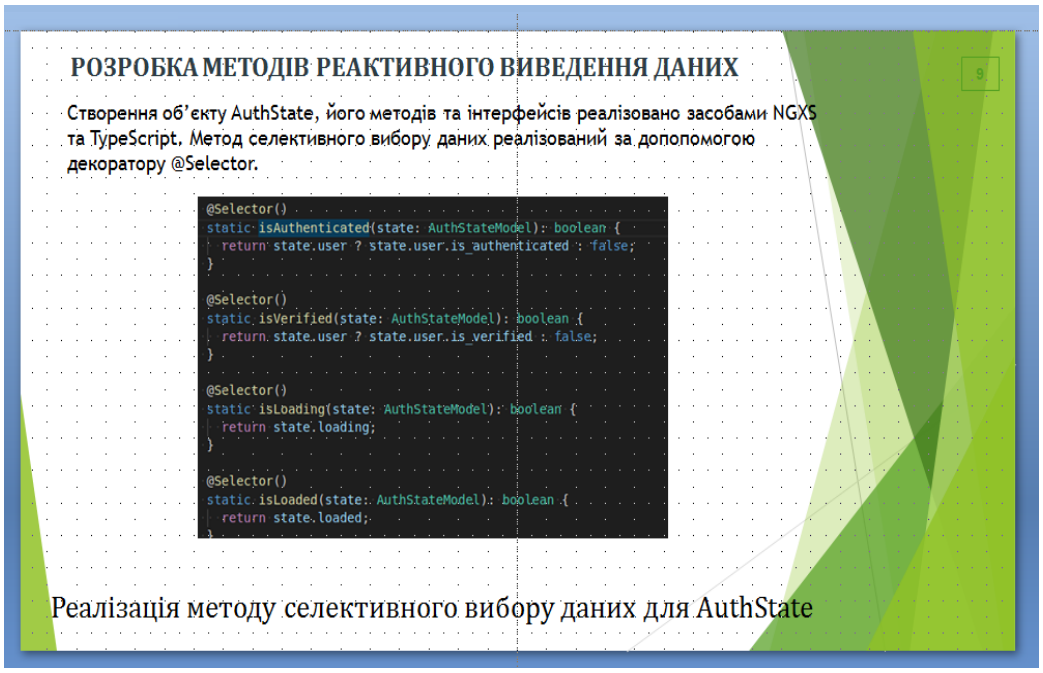


Слайд 7

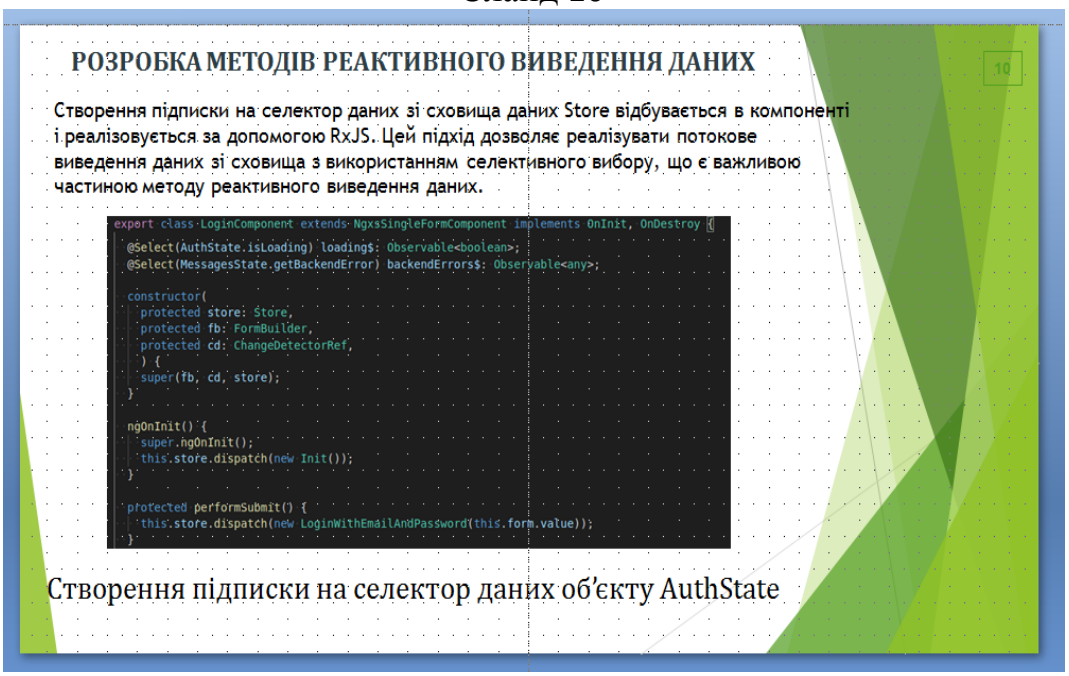
Слайд 8



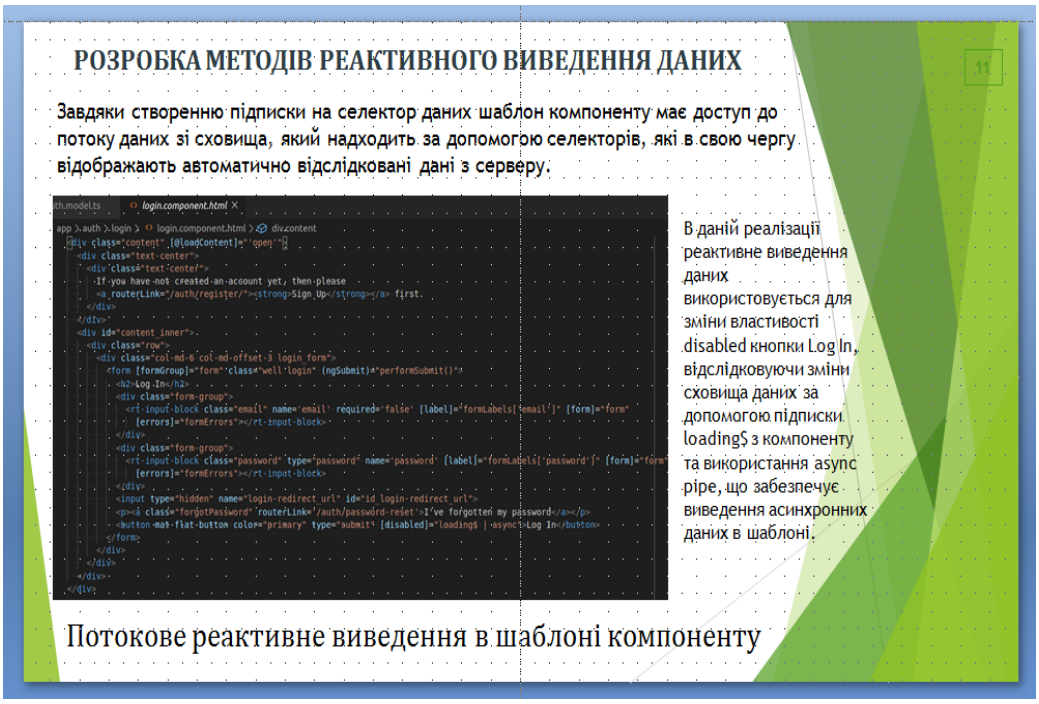
Слайд 9



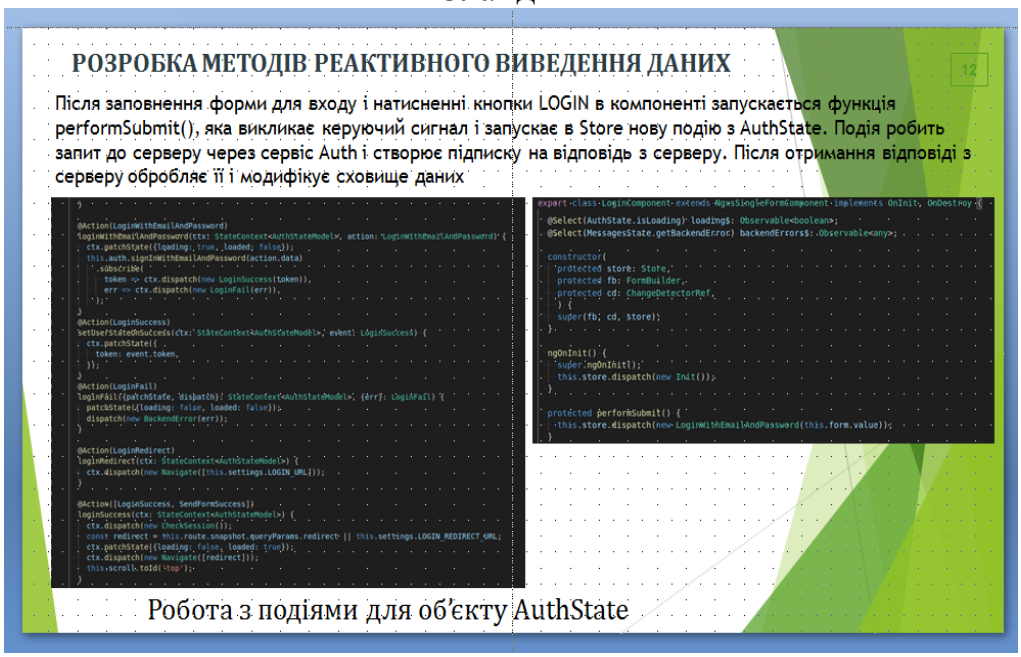
Слайд 10



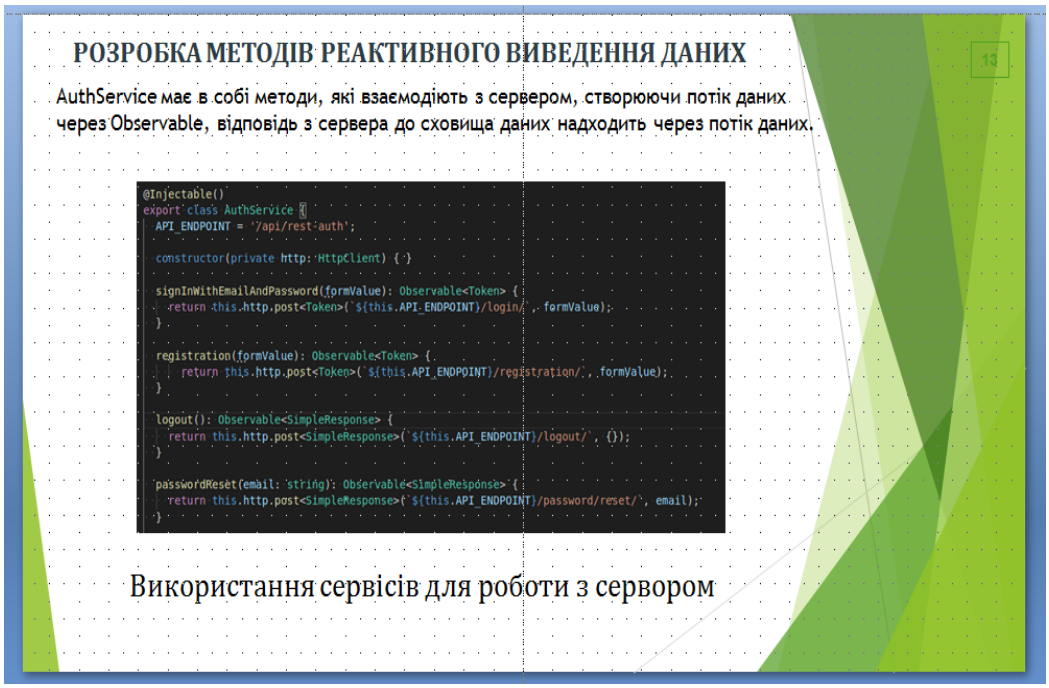
Слайд 11



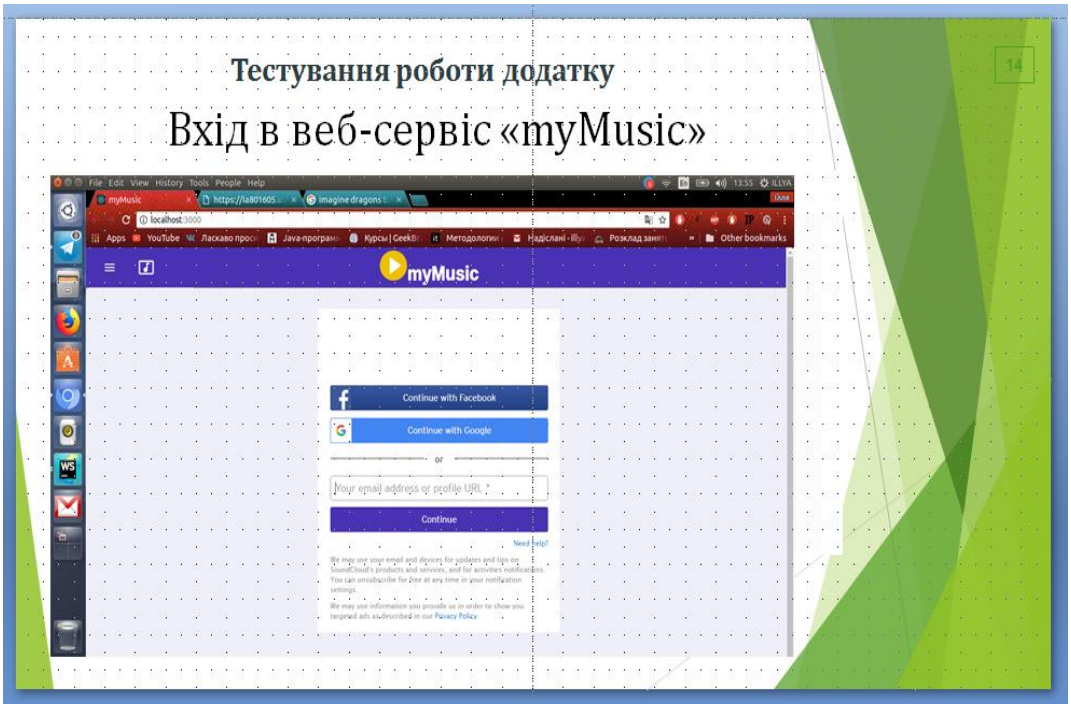
Слайд 12



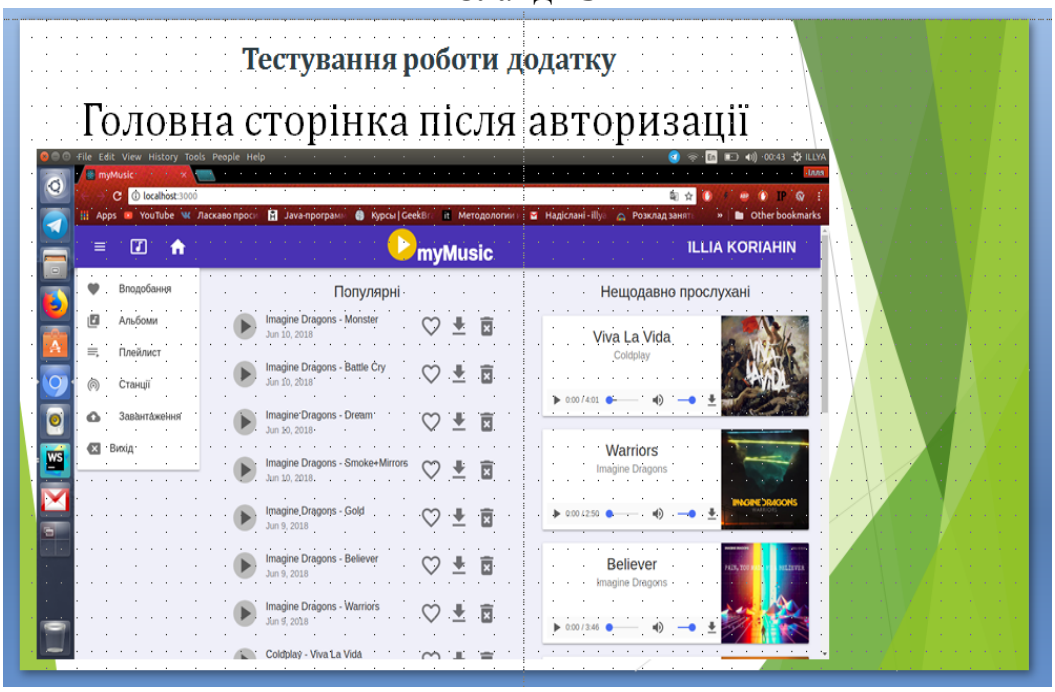
Слайд 13



Слайд 14



Слайд 15



Слайд 16

Слайд 17

ДЯКУЮ ЗА УВАГУ!