МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

«ПОЛТАВСЬКИЙ ПОЛІТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ   
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ   
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Циклова комісія дисциплін програмної інженерії

Затверджую:

Голова циклової комісії

\_\_\_\_\_\_\_\_\_ / Олександр БАБИЧ

“\_\_\_” \_\_\_\_\_\_\_\_\_\_ 2020 р.

**КУРСОВИЙ ПРОЄКТ**

з дисципліни: “Проєктування автоматизованих інформаційних систем”

Розробка автоматизованої інформаційної системи для формування репертуару концерту

на тему \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КП 000.000000.000**

Виконав: здобувач освіти 4 курсу,

44

групи \_\_\_\_\_\_\_  
спеціальність 121 «Інженерія програмного забезпечення»

спеціалізація «Розробка програмного забезпечення»

Лутченко Олена

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище та ініціали)

Світлана ГРИЦЕНКО

Керівник \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 (підпис) (прізвище та ініціали)

Полтава – 2020

ЗМІСТ

[ВСТУП 3](#_Toc55760195)

[1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАВДАННЯ 4](#_Toc55760196)

[1.1. Постановка задачі 4](#_Toc55760197)

[1.2. Опис та побудова діаграми прецедентів роботи з системою 5](#_Toc55760198)

[1.3. Інфологічне проектування. Побудова ER-діаграми 5](#_Toc55760199)

[2. ОПИС ПРОГРАМНИХ ЗАСОБІВ 8](#_Toc55760200)

[3. РЕАЛІЗАЦІЯ РОЗВ’ЯЗАННЯ ЗАДАЧІ ПРОЕКТУВАННЯ 9](#_Toc55760201)

[3.1. Алгоритм програми, архітектура програми 9](#_Toc55760202)

[3.2. Структура програми з функцій складових частин і зв’язків між ними 10](#_Toc55760203)

[3.4. Виклик та завантаження 11](#_Toc55760204)

[3.5. Інструкція користувачу 12](#_Toc55760205)

[ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ 13](#_Toc55760206)

[ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ 15](#_Toc55760207)

[ДОДАТОК Б. ЛІСТИНГ ПРОГРАМИ 19](#_Toc55760208)

# ВСТУП

Створення автоматизованої інформаційної системи для автоматизації створення репертуару концерту є актуальною на даний час. Так як багато музикальних груп роблять юбілейні концерти, на яких вважають за потрібне виконувати тільки найпопулярніші хіти. А також під час презентації нового альбому співають для фанатів старі пісні.

Автоматизована інформаційна система для створення репертуару дозволить покращити і полегшити роботу адміністрації музикальних груп, та завчасно сформувати перелік пісень, котрі фанати найбільше хочуть почути.

Це удосконалить роботу всього персоналу (адміністраторів, креативних директорів та ін.) що входять до команди певної музикальної групи.

*Мета і задачі дослідження:*

* розглянути особливості ведення електронної системи формування репертуару;
* обґрунтувати вибір програмних засобів для розробки повнофункціональної системи «Концерт»;
* спроектувати та розробити автоматизовану систему для формування рейтингу пісень.

# 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАВДАННЯ

## 1.1. Постановка задачі

Автоматизована інформаційна система (АІС) Концерт є основою для автоматизації діяльності будь-якої музикальної групи, що приймає до уваги бажання своїх фанатів. В АІС Концерт повинна оброблятися інформація про пісні та формуватися певний рейтинг, а також запроваджена система реєстрації користувачів та прав адміністратора.

Користувачами такої інформаційної системи можуть бути адміністратори та відвідувачі концерту.

Функціональність АІС:

* реєстрація (додавання, редагування інформації) користувача;
* вибір певної пісні;
* додавання пісні за бажанням;
* можливість змінити користувача;
* формування рейтингу пісень.

Перелік даних, які повинні бути закладені в БД інформаційної системи: «Пісні», «Користувачі», «Користувацькі додатки».

## 1.2. Опис та побудова діаграми прецедентів роботи з системою

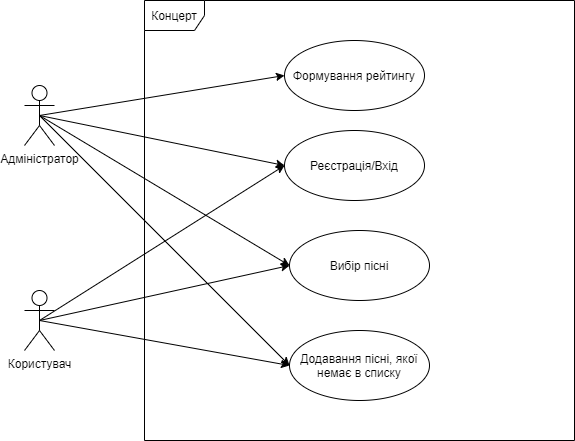


Рисунок 1.1 – Діаграма прецедентів

З програмою може працювати адміністратор та простий користувач. Під користувачем ми розуміємо людину, котра прийшла на концерт. В користувача є можливість авторизуватися (або ввійти до програми, якщо вона вже зареєстрована), вибрати пісні (можна не одну), та додати пісні за власним бажанням. В адміністратора всім можливості аналогічні, але крім цього він може сформувати рейтинг пісень.

## 1.3. Інфологічне проектування. Побудова ER-діаграми

Враховуючи завдання створення ІС, дані що використовуватимуться системою зберігатимуться в базі даних (БД). В БД, доцільно виділити три інформаційних об'єктів (відношень): пісні, користувачі і пісні користувача.

Розглянемо структуру таблиць, що описують ці сутності, і розробимо модель даних «сутність-зв'язок».

Таблиця «Playlist» містить перелік пісень, запропоновані від адміністратора та містить такі поля: «Id», «Title».

У таблиці «User» мають бути такі поля: «Id», «Login», «Password».

Таблиця «UserSong» містить в собі додані пісні від користувача. Атрибути цих таблиць будуть: «Id», «UserId», «SongId».

Такий склад інформаційних об'єктів, включених в БД, дає можливість у результаті обробки інформації, що міститься в таблицях, отримати рейтинг пісень від різних користувачів.

У процесі проектування бази даних можна прийти до висновку, що правильна організація зберігання та подання даних є невід'ємною частиною для успішного функціонування бази даних. Величезну роль у побудові бази даних грають такі корисні функції, як створення запитів, які дозволяють робити вибірку необхідних полів з великою сукупністю даних, а також проводити арифметичні і логічні операції над цими полями.

В результаті виконаної роботи спроектована реляційна база даних, підтримувана СУБД Microsoft SQL Server Compact, в якій міститься три таблиці (Рисунок 1.2).

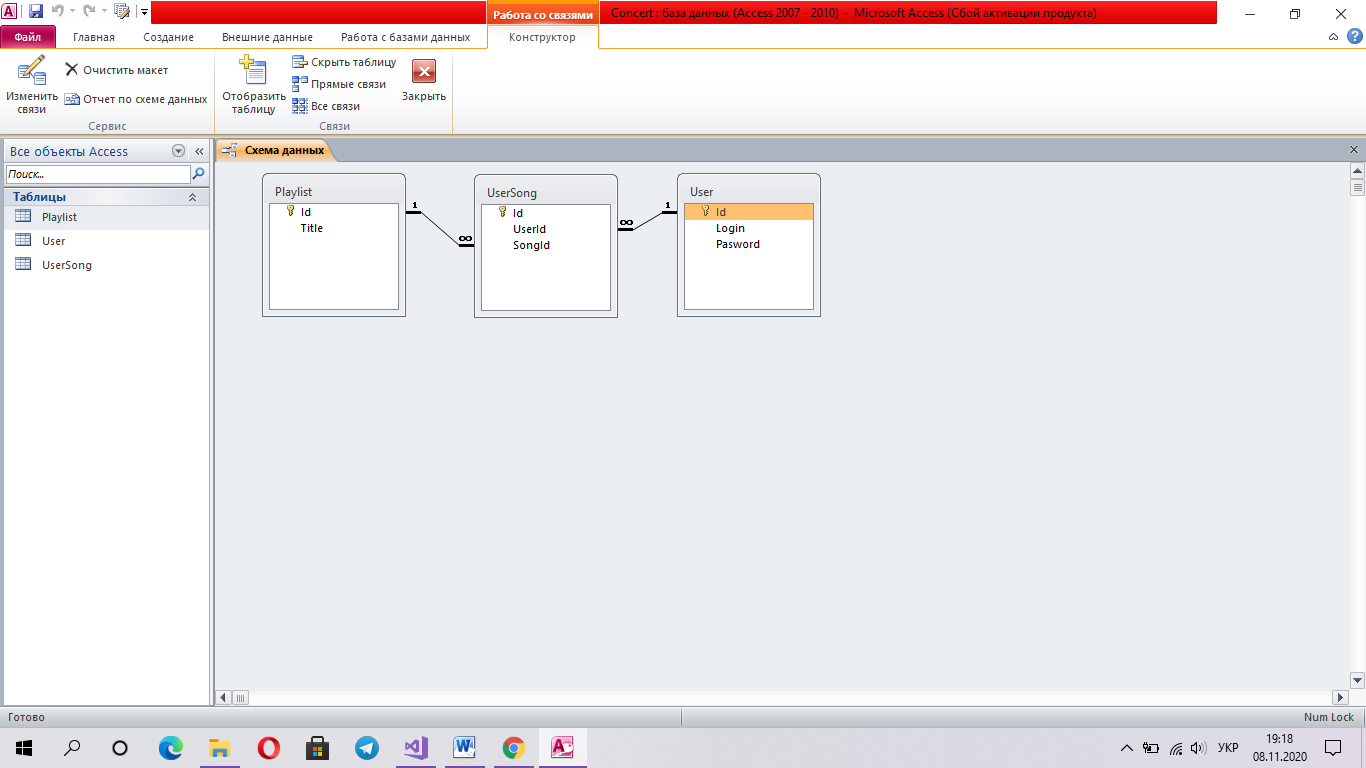


Рисунок 1.2 – Схема бази даних

У цих таблицях містяться відомості про сутності предметної області «Концерт»: пісні, користувачі та ін. Кожна таблиця перевірена на працездатность додатку, розробленого на С# спеціально для адміністрування створеної бази даних.

# 2. ОПИС ПРОГРАМНИХ ЗАСОБІВ

МОВА РОЗРОБКИ

C# - об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Хейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

ПЛАТФОРМА

Оболонка .NET Framework визначає середовище для розробки і виконання сильно розподілених застосувань, заснованих на використанні компонентних об’єктів [14, 15]. Вона дозволяє "мирно співіснувати" різним мовам програмування і забезпечує безпеку, переносимість програм і загальну модель програмування для платформи Windows. .NET Framework по своїй суті не обмежена вживанням в Windows, тобто програми, написані для неї, можна потім переносити в середовища, відмінні від Windows.

СЕРЕДОВИЩЕ РОЗРОБКИ

Microsoft Visual Studio – серія продуктів фірми [Майкрософт](http://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B9%D0%BA%D1%80%D0%BE%D1%81%D0%BE%D1%84%D1%82), які включають [інтегроване середовище розробки](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8) програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як [консольні програми](http://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0), так і програми з [графічним інтерфейсом](http://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D1%96%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), в тому числі з підтримкою технології [Windows Forms](http://uk.wikipedia.org/w/index.php?title=Windows_Forms&action=edit&redlink=1), а також [веб-сайти](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82), [веб-застосунки](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), [веб-служби](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) як в [рідному](http://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4), так і в [керованому](http://uk.wikipedia.org/wiki/%D0%9A%D0%B5%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4) кодах для всіх платформ, що підтримуються[Microsoft Windows](http://uk.wikipedia.org/wiki/Microsoft_Windows), [Windows Mobile](http://uk.wikipedia.org/wiki/Windows_Mobile), [Windows CE](http://uk.wikipedia.org/wiki/Windows_CE), [.NET Framework](http://uk.wikipedia.org/wiki/.NET_Framework), [.NET Compact Framework](http://uk.wikipedia.org/wiki/.NET_Compact_Framework) та [Microsoft Silverlight](http://uk.wikipedia.org/wiki/Microsoft_Silverlight).

# 3. РЕАЛІЗАЦІЯ РОЗВ’ЯЗАННЯ ЗАДАЧІ ПРОЕКТУВАННЯ

## 3.1. Алгоритм програми, архітектура програми

При розробці АІС «Концерт» для реалізації основної частини, яка виконує загальну роботу програми, використовувався алгоритм, наведений на Рисунку 3.1.

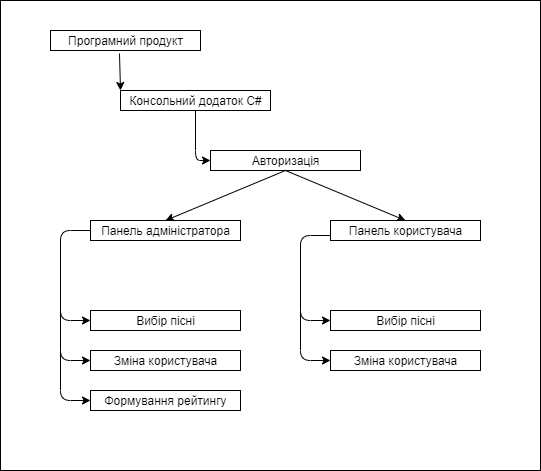


Рисунок 3.1 – Загальна робота програми

Автоматизована інформаційна система «Концерт» представляє собою консольний додаток С#. З додатком може працювати адміністратор та простий користувач. В користувача є можливість авторизуватися (або ввійти до програми, якщо вона вже зареєстрована), вибрати пісню (можна не одну), та додати пісні за власним бажанням. В адміністратора всім можливості аналогічні, але крім цього він може сформувати рейтинг пісень.

## 3.2. Структура програми з функцій складових частин і зв’язків між ними

На рисунку 3.2 представлений скріншот зовнішнього вигляду програми.

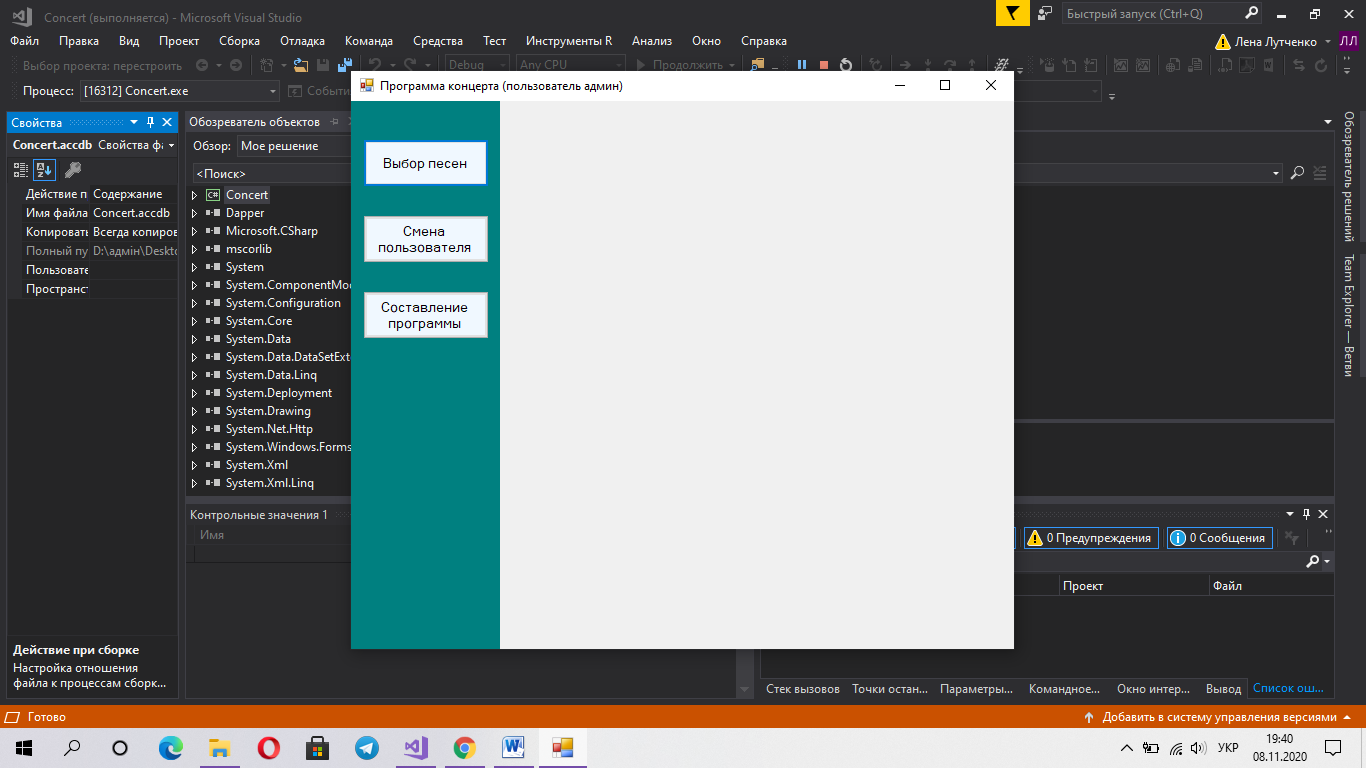


Рисунок 3.2 – Головне меню

На даному скріншоті зображено головне меню програми, що містить три кнопки: «Выбор песен», «Смена пользователя», «Составление программы».

При натисканні кнопки «Выбор песен» в правій частині відкритого вікна з’явиться запропонований список пісень, а також вікно для додавання пісні, котрої немає в списку. (Рисунок 3.3)

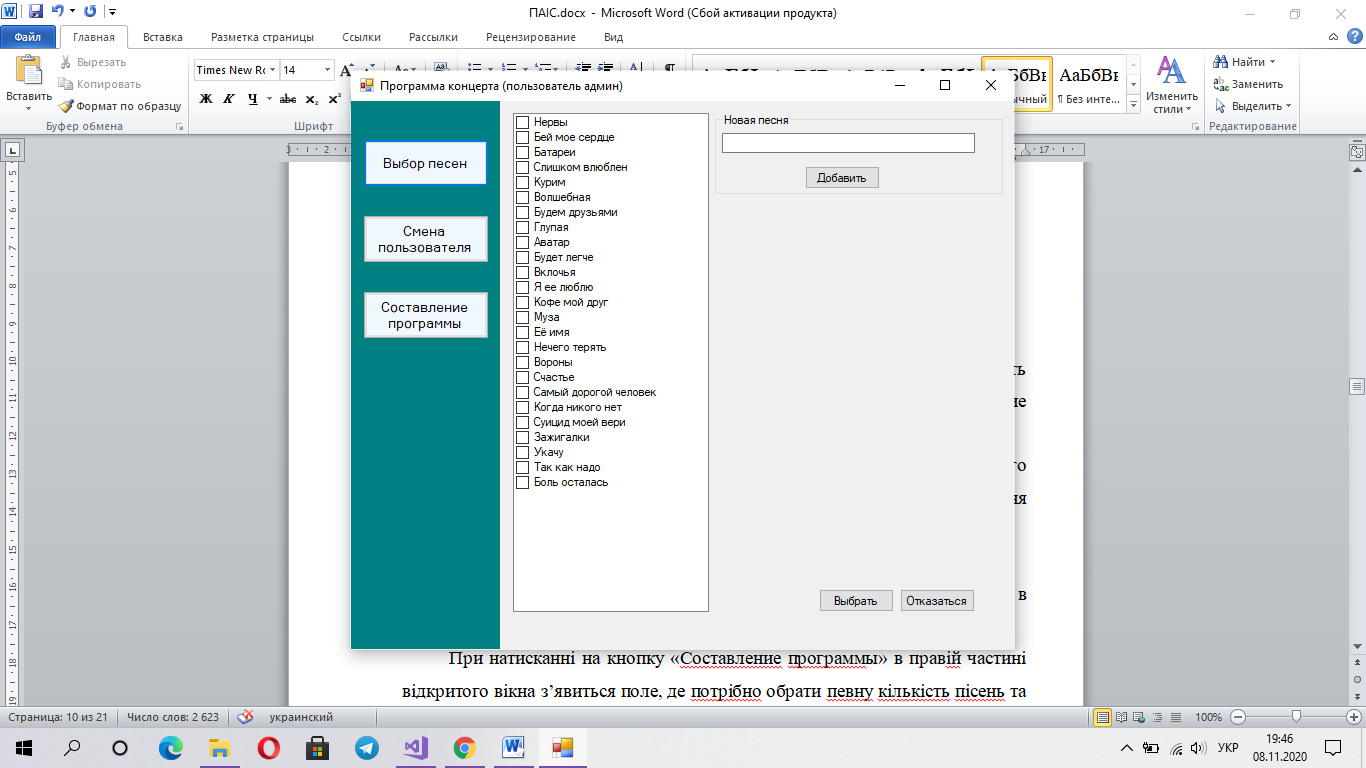


Рисунок 3.3 – «Выбор песен»

При натисканні кнопки «Смена пользователя» відкриється вікно, в якому необхідно авторизуватися/ввійти до додатку. (Рисунок 3.4)

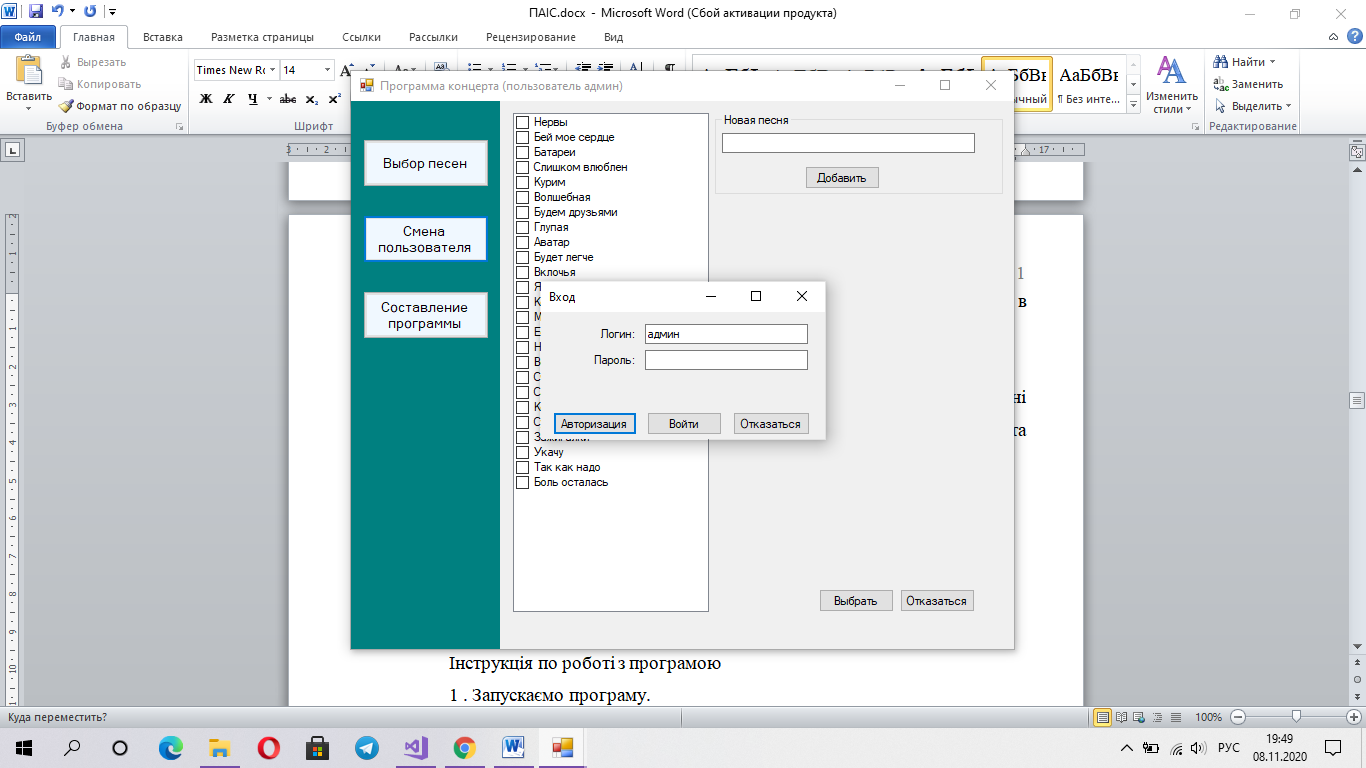


Рисунок 3.4 – «Смена пользователя»

При натисканні на кнопку «Составление программы» в правій частині відкритого вікна з’явиться поле, де потрібно обрати певну кількість пісень та остаточно сформувати список. (Рисунок 3.5)

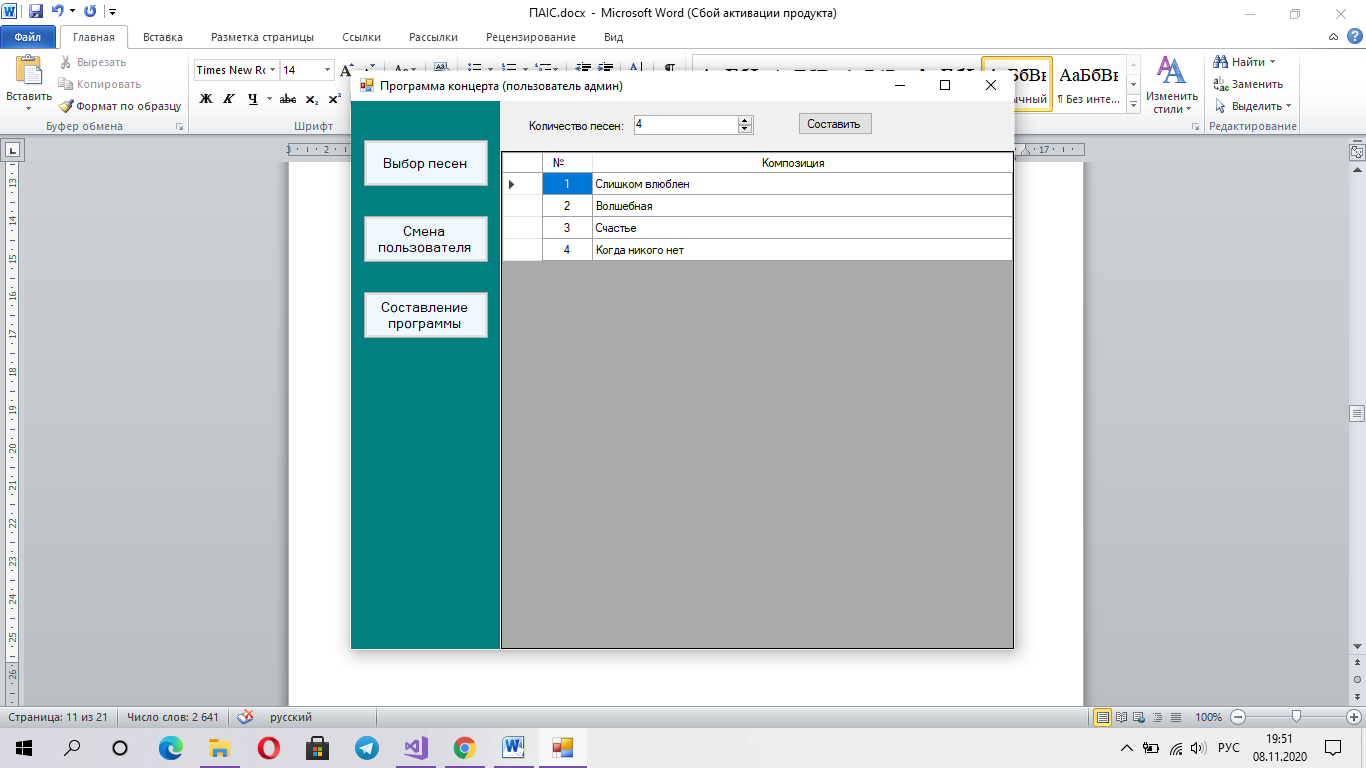


Рисунок 3.5 – «Составление программы»

## 3.4. Виклик та завантаження

Автоматизована система «Концерт» викликається за допомогою відкриття форми С#. Завантаження може відбутися через електронного носія данних, а також через мережу інтернет.

## 3.5. Інструкція користувачу

Інструкція по роботі з програмою для простого користувача.

1. Запускаємо програму.
2. Авторизуємось: вводимо своє ім’я, пароль та підтвердження паролю.
3. Натискаємо на кнопку «Выбор песен»
4. Обираємо потрібну пісню чи додаємо свою та натискаємо кнопку «Добавить».
5. Натискаємо кнопку «Выбрать».

Інструкція по роботі з програмою для простого користувача.

1. Запускаємо програму.
2. Авторизація від імені адміністратора (логін: админ, пароль: 1).
3. Натискаємо на кнопку «Выбор песен»
4. Обираємо потрібну пісню чи додаємо свою та натискаємо кнопку «Добавить».
5. Натискаємо кнопку «Выбрать».
6. Натискаємо кнопку «Составление программы».
7. Обираємо потрібну кількість пісень
8. Натискаємо кнопку «Составить».
9. Завершуємо роботу додатку.

# ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

Було поставлено задачу створити автоматизовану інформаційну систему «Концерт», котра могла б створювати рейтинг пісень за підрахунком кількості голосів користувачів, що полегшувало б роботу адміністраторів музикальних груп.

Робота над цим проєктом повністю виконана та досягнуто всіх очікувань та поставлених завдань.

На мою думку, ця робота є досить актуальною так як додаток можна застосовувати на концерти для формування репертуару або ж просто як спостереження за тенденцією розвитку популярності певних пісень.

Під час створення цього продукту я навчилася правильно формувати поставлене перед мною завдання та створювати відповідну документацію щодо програмного продукту. Я вдосконалила свої навички про роботу з мовою програмування С#, навчилася працювати з базами даних та узагальнила всі свої знання про проектування автоматизованих систем.

В подальшому цей додаток можна розвивати та застосовувати в більш широких колах. Можливо створити додаткові функції по вибору різних груп. Також в майбутньому можна вдосконалити додаток збереженням користувачів в системі після закриття програми та збереженні їх власно доданих композицій.

Створена автоматизована інформаційна система для формування репертуару дозволить покращити і полегшити роботу адміністрації музикальних груп, та завчасно сформувати перелік пісень, котрі фанати найбільше хочуть почути. Це може удосконалить роботу всього персоналу (адміністраторів, креативних директорів та ін.) що входять до команди певної музикальної групи.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Турчин, С. Общие проблемы автоматизации / С. Турчин // Компьютерное обозрение. – 2000. – №6. – С. 48-50.
2. Компьютерные технологии обработки информации / под ред. С.В. Назарова. – М.: Финансы и статистика, 1995. – 247с.
3. Харрингтон, Дж. Л. Проектирование реляционных баз данных : пер. с англ. – М.: Лори, 2006. – 230 с.
4. Базы данных: от проектирования до разроботки приложений /под ред. А.П. Чекалов. – СПб.: БХВ-Петербург, 2003.
5. Ульман, Д. Введение в системы баз данных / Д. Ульман, Д.Уилдом. пер. с англ. – М.: Лори, 2000. – 374 с.
6. Дейт, К. Дж. Введение в системы баз данных: пер. с англ. – М.: Вильямс, 2000. – 848 с.
7. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования : пер. с англ./ Э. Гамма, Р. Хелм, Р Джонсон, Дж. Влиссидес. – СПб.: Питер, 2010. – 368 с.
8. Буч, Г. Объектно-ориентированный анализ и проектирование : пер. с англ. – СПб; М.: Бином, 1999. – 560 с.
9. Изучаем SQL / Джим Киу, Э. Оппель; пер. с англ. Терентьева Д.А. – М.: НТ Пресс, 2007.
10. Oracle PL/SQL : карманный справочник / С. Фейерштейн, Б. Прибыл, Ч. Доз. – СПб.: Питер, 2004.
11. Основы программирования : учеб. пособие / И. Г. Семакин., А.П. Шестаков. – М.: Мастерство, 2002.
12. Visual Studio .NET: разработка приложений баз данных [Текст]. – СПб.: БХВ-Петербург, 2003. – 544 с.: ил.
13. С# и платформа .NET. Библиотека программиста. — СПб.:
14. Полный справочник по С#. Питер, 2004. —796 с.] : пер. с англ. — М.: Вильямс, 2004. — 752 с.

# ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ

1. ПРИЗНАЧЕННЯ І МЕТА СТВОРЮВАНОЇ СИСТЕМИ

1.1. Загальні відомості

Повна назва системи «Автоматизована інформаційна система для формування репертуару концерту». Коротка назва АІС Концерт.

Робота виконується на основі завдання до курсового проекту з дисципліни Проектування автоматизованих інформаційних систем.

Термін виконання роботи: 12 вересня 2020 року – 12 листопада 2020 року.

Роботи по створенню АІС Концерт здаються поетапно згідно з календарним планом.

1.2. Призначення системи

У АІС Концерт повинен формуватися рейтинг пісень.

Функціональність АІС:

* реєстрація (додавання, редагування інформації) користувача;
* вибір певної пісні;
* додавання пісні за бажанням;
* можливість змінити користувача;
* формування рейтингу пісень.

1.3. Мета створення

АІС Концерт створюється з метою забезпечення автоматизованого збору та обробки первинної інформації та створення репертуару до концерту.

2. ХАРАКТЕРИСТИКА ОБ’ЄКТІВ АВТОМАТИЗАЦІЇ ТА ВИМОГИ ДО СИСТЕМИ

2.1. Відомості про об’єкт автоматизації

Об’єктом автоматизації є:

* процес авторизації адміністратора;
* процес реєстрації користувачів;
* процес формування списку пісень за рейтингом.

Особливих умов до експлуатації АІС не висувається

2.2. Вимоги до системи вцілому

АІС повинна бути централізованою, тобто всі дані повинні розташовуватися в центральному сховищі. В Системі пропонується виділити наступні функціональні підсистеми:

* підсистема збору, обробки та завантаження даних;
* підсистема зберігання даних;
* підсистема формування та візуалізації звітності.

2.3. Вимоги до функцій (задач), що виконуються системою

Система повинна забезпечувати

* + роботу користувачів режимі проведення концерту. Тобто декілька годин на день в робочі дні.
  + виконання своїх функцій - збір, обробка та завантаження даних;
  + зберігання даних, надання звітності.

2.4. Вимоги до надійності

Програмний виріб повинен надійно функціонувати в апаратному і програмному середовищі. Відмови, що виникають з-за збоїв апаратних засобів або з-за невідповідності середовища функціонування вимогам, не повинні вести до втрати інформації в пам’яті на магнітних накопичувачах, але припускається ситуація перевантаження операційної системи після відмови.

2.5. Вимоги до інформаційної і програмної сумісності

Для розробки використовувати MS SQL Server Сompact, об’єктоно-орієнтоване середовище Visual Studio 2010, мову програмування C#, платформу Net.Framework 4.5.

2.6. Спеціальні вимоги

Спеціальні вимоги до даного програмного продукту не висуваються.

3. СКЛАД І ЗМІСТ РОБІТ ПО СТВОРЕННЮ СИСТЕМИ

Роботи по створенню системи виконуються в три етапи:

* технічне завдання;
* розробка проекту і проектної документації;
* захист проекту.

На етапі "Технічне завдання" має бути виконаний етап розробки, і етап затвердження технічного завдання:

* постановка завдання;
* визначення і уточнення вимог до технічних засобів;
* визначення вимог до програми;
* визначення стадій, етапів і термінів розробки програми і документації на неї;
* вибір мов програмування;
* узгодження і затвердження технічного завдання.

На етапі "Розробка проекту і проектної документації" мають бути виконані перераховані етапи робіт:

* розробка програми (програмування (кодування) і відладки програми);
* розробка програмної документації (розробка пояснювальної записки, технічного завдання, опису програми відповідно до вимог чинних методичних рекомендацій);
* випробування програми (тестування програми; коригування програми і програмної документації за результатами тестування).

На етапі захисту проекту має бути виконана розробка презентації курсового проекту і представлення його комісії.

4. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ СИСТЕМИ

При прийманні курсового проекту перевіряється – комплектність, зміст і оформлення програмної документації.

Захист проводиться перед комісією з трьох викладачів у присутності студентів поточної академічної групи.

# ДОДАТОК Б. ЛІСТИНГ ПРОГРАМИ

В додатку Б наведено частину програмного коду, що описує ключові моменти роботи програми.

using Dapper;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.OleDb;

using System.Linq;

using System.Windows.Forms;

using Сoncert.Models;

namespace Сoncert

{

public static class DataAcces

{

private static string GetConnectionSting()

{

return ConfigurationManager

.ConnectionStrings["Сoncert.Properties.Settings.ConcertConnectionString"].ConnectionString;

}

public static int Add(PlaylistModel song)

{

int ret = 0;

try

{

using (var conn = new OleDbConnection(GetConnectionSting()))

{

string sql = @"Insert into Playlist

(Title)

VALUES(@Title)";

conn.Execute(sql, song);

sql = @"select MAX(Id)

from Playlist";

ret = conn.Query<int>(sql).Single();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return ret;

}

public static int AddUser(UserModel user)

{

int ret = 0;

try

{

if (LoginCheck(user))

throw new Exception("Логин уже существует.");

using (var conn = new OleDbConnection(GetConnectionSting()))

{

string sql = @"Insert into [User]

(Login ,Pasword)

VALUES(@Login ,@Pasword)";

conn.Execute(sql, user);

sql = @"select MAX(Id)

from [User]";

ret = conn.Query<int>(sql, new DynamicParameters()).Single();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return ret;

}

public static int AddUserSong(UserSongModel userSong)

{

int ret = 0;

try

{

using (var conn = new OleDbConnection(GetConnectionSting()))

{

string sql = @"Insert into UserSong

(SongId,UserId)

VALUES(@UserId,@SongId)";

conn.Execute(sql, userSong);

sql = @"select MAX(Id)

from UserSong";

ret = conn.Query<int>(sql).Single();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return ret;

}

public static int CheckUser(UserModel user)

{

int ret = 0;

try

{

using (var conn = new OleDbConnection(GetConnectionSting()))

{

string sql = @"Select \*

from [User]

Where Login =@Login and Pasword=@Pasword";

var output = conn.Query<UserModel>(sql, user).FirstOrDefault();

if (output != null)

ret = output.Id;

else

throw new Exception("Пароль неверный");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return ret;

}

public static void ClearSongFromUser(int userId)

{

try

{

using (var conn = new OleDbConnection(GetConnectionSting()))

{

string sql = @"Delete from UserSong

Where (UserId =@UserId )";

var p = new DynamicParameters();

p.Add("@UserId", userId);

conn.Execute(sql, p);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

public static List<PlaylistModel> GetPlaylist()

{

var list = new List<PlaylistModel>();

try

{

using (var conn = new OleDbConnection(GetConnectionSting()))

{

string sql = "Select \* from Playlist";

var output = conn.Query<PlaylistModel>(sql, new DynamicParameters());

list.AddRange(output);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return list;

}

public static List<UserSongModel> GetUserSongs()

{

var list = new List<UserSongModel>();

try

{

using (var conn = new OleDbConnection(GetConnectionSting()))

{

string sql = "Select \* from UserSong";

var output = conn.Query<UserSongModel>(sql, new DynamicParameters());

list.AddRange(output);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return list;

}

public static List<UserSongModel> GetUserSongs(UserModel user)

{

var list = new List<UserSongModel>();

try

{

using (var conn = new OleDbConnection(GetConnectionSting()))

{

string sql = @"Select \* from UserSong

where UserId=@Id";

var output = conn.Query<UserSongModel>(sql, user);

list.AddRange(output);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return list;

}

public static bool LoginCheck(UserModel user)

{

bool ret = true;

try

{

using (var conn = new OleDbConnection(GetConnectionSting()))

{

string sql = @"Select count(\*)

from [User]

Where (Login =@Login )";

ret = conn.Query<int>(sql, user).Single() > 0;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return ret;

}

}

}