Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

**ПРАКТИЧЕСКАЯ РАБОТА**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Тема: **«Программная реализация КИС»**

Выполнил:

студент гр. . ИСТ-19-1б .

. Дорогина Ирина Валерьевна .

(Ф.И.О.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подпись)*

Принял:

ст. преподаватель, Банников Р.Ю.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(должность, ФИО руководителя)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Пермь 2023**

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc128160304)

[2 Реализация информационной системы 3](#_Toc128160305)

[2.1 Средства разработки 3](#_Toc128160306)

[2.2 Функционал 3](#_Toc128160307)

[Вывод 7](#_Toc128160308)

[Приложение 1. Листинг 1. Редактирование информации базы данных 8](#_Toc128160310)

1 Постановка задачи

Информационная системы «Хранилище данных музыкальной платформы “Spot”» предназначена для комплексного информационно-аналитического обеспечения в части исполнения следующих процессов:

* формирование и сопровождение перечня представленных на музыкальной платформе «Spot» композиций;
* формирование и сопровождение информации (название композиции, исполнитель, название альбома, жанр) о каждой композиции;
* формирование и сопровождение перечня слушателей (имя, дата регистрации, страна) музыкальной платформы «Spot»;
* формирование и сопровождение информации о прослушиваемых пользователями плейлистах, составленных из композиций, представленных на музыкальной платформе «Spot».

Описываемую информационную систему предполагается использовать для регулирования управления музыкальной площадкой посредством анализа предпочтений слушателей и популярности треков на основе имеющихся в перечнях данных.

В состав информационной системы должны входить:

1. Перечень «Композиции»;
2. Перечень «Слушатели»;
3. Перечень «Плейлисты»;
4. Перечень «Пользователи системы» (отображается только у администраторов системы).

Допускается наличие дополнительных перечней.

Программа должна обеспечивать возможность составления отчетов на основании имеющихся в перечнях базы данных системы данных:

1. отчет «Рейтинг популярности музыкальных жанров среди слушателей»;
2. отчет «Рейтинг плейлистов среди слушателей»;
3. отчет «Рейтинг стран по наибольшему числу слушателей»;
4. отчет «Самые популярные композиции в каждом жанре»;
5. отчет «Список песен каждого музыкального жанра».

2 Реализация информационной системы

2.1 Средства разработки

Для разработки информационной системы был выбран язык программирования C#, поэтому обоснованным является выбор платформы пользовательского интерфейса для создания классических приложений Windows – Windows Form в среде разработки Visual Studio.

База данных для информационной системы реализована с помощью объектно-реляционной СУБД PostgreSQL.

2.2 Функционал

Рассмотрим по порядку основные функциональные составляющие информационной системы «Хранилище данных музыкальной платформы “Spot”», которые было необходимо реализовать:

1. Авторизации пользователя;

2. Основная страница приложения для пользователя с правами администратора;

3. Основная страница приложения для пользователя с правами аналитика;

4. Редактирование информации в базе данных;

5. Формирование отчётов;

6. Реакция информационной системы на ошибочный ввод данных.

В рамках функционала авторизации стояла задача сравнения данных, введённых пользователем, с имеющимися в базе данных.

В DataBaseManager все, полученные из базы данных записи по пользователям системы, проверяются на соответствие предоставленным логину и паролю. Если пользователь введёт логин и (или) пароль, не зарегистрированные в системе или введённые с ошибкой, то программа проинформирует его об этом (рисунок 2).

|  |  |
| --- | --- |
|  |  |
| Рисунок 1 – Интерфейс формы для авторизации пользователя | Рисунок 2 – Сообщение об ошибке авторизации |

При успешной авторизации пользователь будет попадать в основное окно приложения, соответствующее его уровню доступа. Реализация анализа соответствия введённых логина и пароля имеющимся в базе данных на стороне страницы авторизации представлена листингом 1.

|  |
| --- |
| private void loginButton\_Click(object sender, EventArgs e){  DataBaseManager dbManager = new DataBaseManager();  String[] userInfo = dbManager.UserAuthorize(userNameText.Text, userPasswordText.Text).Split(' ');  if(userInfo[1] == "administrator"){  this.Hide();  AdminForm adminForm = new AdminForm(userInfo[0]);  adminForm.Show();  }  else if(userInfo[1] == "manager"){  this.Hide();  UserForm userForm = new UserForm(userInfo[0]);  userForm.Show();  }  else{  DialogResult result = MessageBox.Show("Неправильный логин или пароль", "Ошибка входа",MessageBoxButtons.OK, MessageBoxIcon.Error, MessageBoxDefaultButton.Button1, MessageBoxOptions.RightAlign);  if (result == DialogResult.OK){  userNameText.Clear();  userPasswordText.Clear();  }  } |
| Листинг 1 – Авторизация пользователя |

На рисунке 3(а, б) представлены интерфейсы основного окна приложения для пользователя с уровнем доступа Аналитик и Администратор, соответственно. Так для администратора функционал приложения расширяется, за счёт наличия инструментов для просмотра и редактирования информации из базы данных информационной системы.

|  |
| --- |
| а) |
| б) |
| Рисунок 3 – Интерфейс основного окна: а) для аналитика; б) для Администратора |

В окне (рисунок 3б) элементы сгруппированы по функционалу:

1. Группа для работы с данными из базы данных – выбор таблицы, которую необходимо показать, элемент DataGridView (серое поле слева) для отображения данных, кнопка для обновления таблицы.
2. Группа для формирования отчётов (аналогична функционалу основного окна для Аналитика (рисунок 3а)).

Редактирование информации в таблицах базы данных реализовано с помощью отслеживания событий, связанных с элементом DataGridView. Данный компонент позволяет динамически добавлять строки в таблицу и вводить в них данные. В приложении 1 листинг демонстрирует функционал по обработке событий:

1. Нажатие на содержимое ячейки (dataBaseGridView\_CellContentClick);

2. Изменение значения ячейки (dataBaseGridView\_CellValueChanged);

3. Завершение добавления строки в элемент управления DataGridView (dataBaseGridView\_UserAddedRow).

Удаление строки из таблицы осуществляется с помощью ссылочной надписи Delete (рисунок 3б). При нажатии на неё в DataBaseManager отсылается название изменяемой таблицы и строка, которую необходимо удалить. Там в базу данных направляется соответствующий запрос, например, «DELETE FROM user\_system WHERE user\_login = '{userInfo[0]}' AND user\_password = '{userInfo[1]}», где userInfo – строка для удаления, формируемой на языке SQL.

Добавление новой записи в таблицу осуществляется через ввод данных в пустую строку отображаемой таблицы. При этом ссылочная надпись Delete заменяется на Insert. Далее, аналогично удалению, в DataBaseManager направляется необходимая информация и в базу данных отправляется сформированный запрос, например, «INSERT INTO genre (genre\_name) VALUES ('{userInfo[0]}')».

Аналогичен процесс обновления уже существующей записи в отображаемой таблице. Пример запроса в базу данных – «UPDATE genre SET genre\_name = '{userInfo[0]}' WHERE genre\_id={editingRowIndex}».

После изменения данных в базе данных обновлённая таблица загружается в элемент DataGridView.

Для формирования отчётов предусмотрены кнопки, которые инициализируют обработку данных из базы данных информационной системы и их запись в текстовый документ формата MS Word. На рисунке 4 представлен пример отчёта «Рейтинг популярности музыкальных жанров среди слушателей».

|  |
| --- |
|  |
| Рисунок 4 – Пример отчёта  «Рейтинг популярности музыкальных жанров среди слушателей» |

Для создания и заполнения документа MS Word было использовано пространство имён Microsoft.Office.Interop.Word. Полученная с помощью DataBaseManager из базы данных таблица, соответствующая полученному запросу на определённый отчёт, направляется в ReportManager, где с помощью метода CreateReport() формируется сам документ.

Если при формировании отчёта произошёл какой-либо сбой, то будет выброшено окно об ошибке (рисунок 5а). Если же процесс создания и заполнения документа прошёл успешно, то пользователь увидит соответствующее сообщение (рисунок 5б).

|  |  |
| --- | --- |
| а) |  |
| Рисунок 5 – Уведомление пользователя о завершении процесса формирования отчёта:  а) при ошибке; б) при успехе | |

В рамках реакции программы на неверно введённые данные предусмотрены описанные выше сообщения об ошибках (рис. 2 и 5а).

Вывод

В рамках программной реализации информационной системы были разработаны необходимые окна приложения, класс для взаимодействия с базой данных и отдельный класс для работы с отчётами. Согласно сформулированному техническому заданию весь необходимый функционал приложения осуществлён.

На следующем этапе работы над информационной системой «Хранилище данных музыкальной платформы “Spot”» планируется провести ручное тестирование для выявления слабых мест программы и проверки корректности её работы с базой данных.

Приложение 1

Листинг 1. Редактирование информации базы данных

private void dataBaseGridView\_CellContentClick(object sender, DataGridViewCellEventArgs e){

try{

if(e.ColumnIndex == dataBaseGridView.Columns.Count-1){

String task = dataBaseGridView.Rows[e.RowIndex].Cells[dataBaseGridView.Columns.Count - 1].Value.ToString();

if(task == "Delete"){

String data = "";

for(int i=0; i<dataBaseGridView.Columns.Count; i++){

data += dataBaseGridView.Rows[e.RowIndex].Cells[i].Value.ToString();

if (i!=dataBaseGridView.Columns.Count-2) data+=" ";

}

dbManager.DeleteEntry(currentTableNameLabel.Text, data);

ReloadTable(currentTableNameLabel.Text);

}

else if(task == "Insert"){

newRowAdding = false;

dataBaseGridView.Rows[e.RowIndex].

Cells[dataBaseGridView.Columns.Count-1].Value="Delete";

String data = "";

for(int i=0;i<dataBaseGridView.Columns.Count-1; i++){

data += dataBaseGridView.Rows[e.RowIndex].Cells[i].Value.ToString();

if(i!=dataBaseGridView.Columns.Count-2) data+=" ";

}

dbManager.InsertEntry(currentTableNameLabel.Text, data);

ReloadTable(currentTableNameLabel.Text);

}

else if(task == "Update"){

dataBaseGridView.Rows[e.RowIndex].

Cells[dataBaseGridView.Columns.Count-1].Value="Delete";

String data = "";

for (int i=1; i<dataBaseGridView.Columns.Count-1; i++){

data += dataBaseGridView.Rows[e.RowIndex].Cells[i].Value.ToString();

if(i!=dataBaseGridView.Columns.Count-2) data+=" ";

}

dbManager.UpdateEntry(currentTableNameLabel.Text, e.RowIndex, data);

ReloadTable(currentTableNameLabel.Text);

}

}

}

catch (Exception ex){

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataBaseGridView\_UserAddedRow(object sender, DataGridViewRowEventArgs e){

try{

if(!newRowAdding){

newRowAdding = true;

int lastRow = dataBaseGridView.Rows.Count - 2;

DataGridViewRow row = dataBaseGridView.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataBaseGridView[dataBaseGridView.Columns.Count - 1, lastRow] = linkCell;

row.Cells[dataBaseGridView.Columns.Count - 1].Value = "Insert";

}

}

catch(Exception ex){

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataBaseGridView\_CellValueChanged(object sender, DataGridViewCellEventArgs e){

try{

if(!newRowAdding){

int rowIndex = dataBaseGridView.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dataBaseGridView.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataBaseGridView[dataBaseGridView.Columns.Count - 1, rowIndex] = linkCell;

editingRow.Cells[dataBaseGridView.Columns.Count - 1].Value = "Update";

}

}

catch (Exception ex){

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}