Министерство образования и науки Украины

Харьковский национальный университет радиоэлектроники

Факультет Автоматики и компьютеризованных технологий

Кафедра Системотехники

**КУРСОВА РОБОТА**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

тема: Разработка элементов информационной системы «Автовокзал»

(тема работы)

по дисциплине Системы управления базами данных

(название дисциплины)

Руководитель доц.каф.СТ Колесник Л.В.

(подпись, дата, должность, фамилия, инициалы)

Студент СИ-14-2 Беседин Б.В.

(группа, подпись, дата, фамилия, инициалы)

Работа защищена оценкой «\_\_\_\_\_\_\_\_\_»

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Комиссия:

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

(подпись, должность, фамилия, инициалы)

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

(подпись, должность, фамилия, инициалы)

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

(подпись, должность, фамилия, инициалы)

Харьков 2016

Харьковский национальный университет радиоэлектроники

Факультет Автоматики и компьютеризованных технологий

Кафедра Системотехники

Дисциплина Системы управления базами данных

Направление Системная инженерия

Курс 2 группа СИ-14-2 семестр 4

##### ЗАДАНИЕ

на курсовую работу

студенту Беседину Богдану Владимировичу

(фамилия, имя, отчество)

1. Тема работы Разработка элементов информационной системы

«Автовокзал»

2. Срок сдачи студентом законченной работы

3. Выходные данные к проекту разработка информационного обеспечения системы администрирования автовокзала. Перечень использованных программных средств: Microsoft Visual Studio 2010, OC Microsoft Windows 10, MySQL Workbench,

4. Содержание пояснительной записки (перечень вопросов подлежащих рассмотрению) 4.1 Введение. 4.2 Описание предметной области. 4.3 Проектирование БД. 4.3.1 ER-моделирование. 4.3.1.1 Описание сущностей. 4.3.1.2 Описание связей. 4.3.2 Разработка физической модели базы данных. 4.3.3 Нормализация базы данных. 4.3.4 Выбор СУБД. 4.4 Выводы. 4.5 Перечень ссылок 4.6 Приложение А. Руководство пользователя.

Перечень графического материала 5.1 Схема базы данных

6. Дата выдачи задания: 22.02.2016

Руководитель работы

(подпись) (фамилия, имя, отчество)

Студент

(подпись) (фамилия, имя, отчество)

Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название этапов курсовой работы | Срок исполнения | Примечание |
| 1. | Получение задание на курсовую работу | 22.02.2016 |  |
| 2. | Анализ предметной области | \_\_.\_\_.\_\_ |  |
| 3. | Разработка модели «сущность-связь» | \_\_.\_\_.\_\_ |  |
| 4. | Проектирование БД | \_\_.\_\_.\_\_ |  |
| 5. | Выбор СУБД | \_\_.\_\_.\_\_ |  |
| 6. | Разработка программного средства | \_\_.\_\_.\_\_ |  |
| 7. | Разработка документа «Руководство пользователя» | \_\_.\_\_.\_\_ |  |
| 8. | Оформление пояснительной записки | \_\_.\_\_.\_\_ |  |
| 9. | Защита курсовой работы | 06.06.2016 |  |

Студент *Беседин Б.В.*

(подпись) (фамилия, имя, отчество)

Руководитель работы *Колесник Л.В*.

(подпись) (фамилия, имя, отчество)

«\_\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_г.

РЕФЕРАТ

Пояснительная записка содержит 21 страницу, 3 таблицы, 5 рисунков, 3 приложения.

Объект исследования - информационная система управления автовокзалом.

Цель курсовой работы – разработка информационной системы для диспетчера, который управляет всеми структурами автовокзала.

Методы исследования – изучение литературы, связанной с данной задачей; исследование реальных БД, для создания прототипа системы «Автовокзал»; разработка ER-модели и физической схемы базы данных с помощью SQL-запросов; написание прикладной программы.

Результат курсовой работы – разработанное программное обеспечение, которое позволяет вносить изменения, добавлять или удалять данные диспетчеру по всей системе «Автовокзал».

Область применения – может использоваться как программное средство диспетчерами автовокзалов, которые имеют доступ к данной программе.

ДИСПЕТЧЕР АВТОВОКЗАЛА, АВТОБУС, БИЛЕТ, ПАССАЖИР, ВОДИТЕЛЬ, РЕЙС, СУБД, БАЗА ДАННЫХ, ER-МОДЕЛЬ, MYSQL.

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ……………………………………………………………………7

2 ОПИСАНИЕ ПРЕДМЕНТНОЙ ОБЛАСТИ………………………………8

3 ПРОЕКТИРОВАНИЕ БД…………………………………………………..9

3.1 ER-моделирование………………………………………………....9

3.1.1 Описание сущности…………………………………….....9

3.1.2 Описание связей………………………………………... ..10

4 ВЫБОР СУБД И СРЕДЫ ПРОГРАММИРОВАНИЯ…………………...14

ВЫВОДЫ……………………………………………………………….……16

ПЕРЕЧЕНЬ ССЫЛОК………………………………………………………17

ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ……………………………….18

ПРИЛОЖЕНИЕ Б. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ………………...47

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ, ЕДИНИЦ, СОКРАЩЕНИЙ И ТЕРМИНОВ

СУБД – система управления базами данных.

БД – база данных.

1 ВВЕДЕНИЕ

В современном мире роль базы данных достаточна высока. Многие предприятия, фирмы используют такой метод хранения информации в компьютере, будь то данные о сотрудниках, о различных коммерческих сделках (покупки, продажи и т.д.).

Система управления базой данных (СУБД) – это программа, с помощью которой в компьютер вводится информация, просматривается, сортируется, фильтруется, разыскивается, экспортируется (переводится в форматы других СУБД) или, наоборот, импортируется. СУБД это программа, которая осуществляет еще и наиболее быстрое обращение к хранящимся в ней данным.

Сегодня большинство систем управления базами данных (СУБД) позволяют размещать в своих структурах не только данные, но и методы (то есть программный код), с помощью которых происходит взаимодействие с потребителем или с другими программно-аппаратными комплексами.

В СУБД имеется возможность отбора, сортировки отображаемых данных в соответствии с заданным критерием, их оформление и последующая выдача на устройство вывода или передача по каналам связи.

Проектируемая база данных предназначена для информационной системы (ИС) диспетчеров автовокзала, для управления и учёта выездов всех автобусов. База данных должна решать довольно узкий круг задач, связанный с сопоставлением расписания и фактических выездов автобусов по различным маршрутам. Автовокзал является промежуточным звеном между другими автовокзалами и пассажирами. Наличия этого звена выгодно и тем и другим: автовокзалы объединены в единую сеть с возможностью взаимной реализации билетов и передачи справочной информации; пассажиры, с другой стороны, не имеют проблем с покупкой билетов на тот или другой авторейс.

2 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Эффективное функционирование современного предприятия невозможно без применения информационных систем. Данная проблема актуальна как для крупных предприятий, так и для предприятий среднего и даже малого бизнеса. Информационные системы имеют ряд существенных отличий от стандартных прикладных программ. В зависимости от предметной области информационные системы могут сильно различаться по своей архитектуре и функциям.

При разработке базы данных «Автовокзал» было проведено обследование предметной области.

Основными операциями в исследуемой области являются составления расписания движения пассажирских автобусов.

Автобусы отправляются по различным маршрутам с различных автостанций. Большое количество маршрутов, частое отправление автобусов заставляет тратить много времени на составление расписания, поэтому основной целью данного курсового проекта является автоматизировать весь этот процесс, чтобы сократить время оператора на обработку данных.

3 ПРОЕКТИРОВАНИЕ БД

3.1 ER-моделирование

Моделирование данных – это первый шаг на пути проектирования БД, это переход от объектов реального мира к компьютерной модели БД. ER-модель служит для объединения различных представлений данных на концептуальном уровне. На основе ER-модели строятся ER-диаграммы, на которых отображаются три основных компонента ER-модели: сущности, атрибуты, связи.

ER-модель представляет собой формальную конструкция, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма сущность-связь (ER-диаграма).

Понятия ER-модель и ER-диаграмма часто ошибочно не различают, хотя для визуализации ER-моделей предложены и другие графические нотации.

3.1 Описание сущностей

Проанализировав предметную область и приняв во внимание основные функции разрабатываемого программного продукта и список входящих документов можно выделить следующие сущности, представленные в таблице 2.1

Таблица 3.1.1 – Сущности и их атрибуты

|  |  |
| --- | --- |
| Название сущности | Описание |
| Buses | idBus, idDriver, Company, Model, Passengers |
| Flights | idFlight, idBus, Departure\_time, Arrival\_time, Departure\_place, Arrival\_place, Intermediate\_stops |
| Drivers | IdDriver, Surname, Name, Patronymic, Wage |
| Passengers | idPassenger, Surname, Name, Patronymic, Nationality |
| Tickets | idTicket, idPassenger, idFlight, idBus |

3.1.2 Описание связей

Таблица 3.2 – Описание связей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сущность 1 | Атрибут 1 | Наименование | Кратность | Сущность 2 | Атрибут 2 |
| Flight | idBus | Имеет | 1:1 | Bus | idBus |
| Bus | idDriver | Имеет | 1:1 | Driver | idDriver |
| Tickets | idPassenger | Имеет | 1:1 | Passenger | idPassenger |
| Tickets | idFlight | Имеет | M:1 | Flight | idFlight |
| Tickets | idBus | Имеет | M:1 | Bus | idBus |

1. В одном рейсе может быть задействован лишь один автобус.
2. Один водитель может управлять одним автобусом.
3. Один билет может принадлежать одному и только одному пассажиру.
4. На один рейс может быть продано множество билетов.
5. На один автобус может быть продано множество билетов.

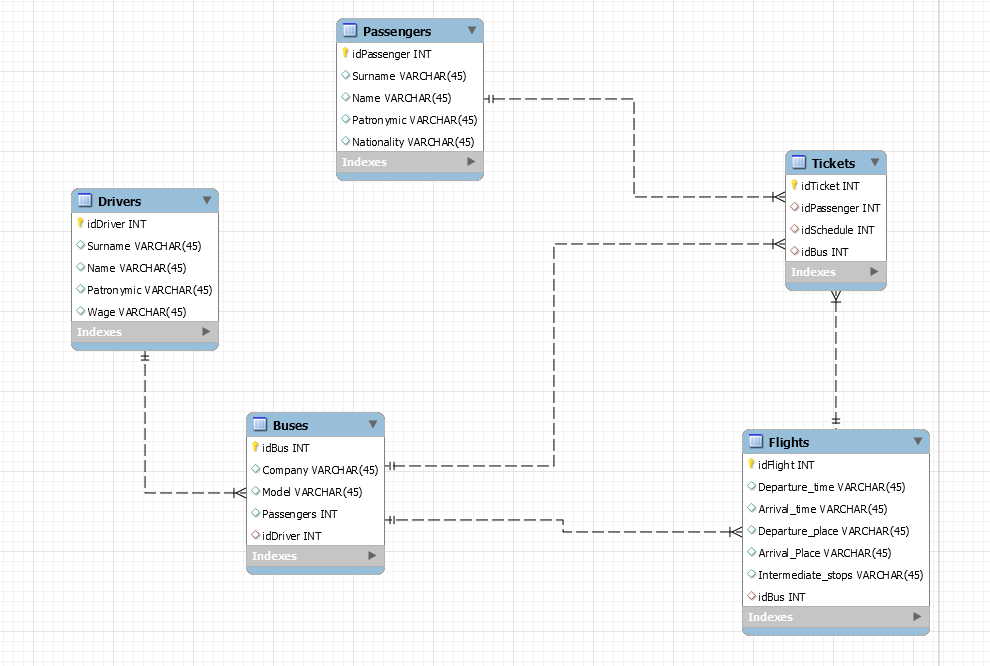


Рисунок 3.1- Обобщенная ER-модель

3.2 Нормализация разработанной БД

Приведение таблиц к первой нормальной форме:

Таблицы, представленные в схеме базы данных, приведены в первой нормальной форме, так как в них определены все ключевые атрибуты, все значения атомарные, все атрибуты зависят от первичного ключа.

Приведение таблиц ко второй нормальной форме:

Таблицы, представленные в схеме базы данных, приведены ко второй нормальной форме, так как они приведены в первой нормальной форме и в них отсутствуют частичные зависимости.

Приведение таблиц к третьей нормальной формы:

Таблицы, представленные в схеме базы данных, приведены в третьей нормальной форме, так как они приведены ко второй нормальной форме и отсутствуют транзитивные зависимости.

Схемы данных таблиц, приведенных на рисунках 2.2 – 2.9.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| idBus | Company | Model | Passengers | idDriver |

Рисунок 3.2 – Диаграмма зависимостей таблицы "Buses"

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| idFlight | Departure\_time | Arrival\_time | Departure\_place | Arrival\_place | Intermediate\_stops | idBus |

Рисунок 3.3 – Диаграмма зависимостей таблицы "Flights"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| idDriver | Surname | Name | Patronymic | Wage |

Рисунок 3.4 – Диаграмма зависимостей таблицы "Drivers"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| idPassenger | Surname | Name | Patronymic | Nationality |

Рисунок 3.5 – Диаграмма зависимостей таблицы "Passengers"

|  |  |  |  |
| --- | --- | --- | --- |
| idTickets | idPassenger | idFlight | idBus |

Рисунок 3.6 – Диаграмма зависимостей таблицы "Tickets"

4 ВЫБОР СУБД И СРЕДЫ ПРОГРАММИРОВАНИЯ

Выбор СУБД является одним из главных решений для разработчика базы данных, поскольку от выбора СУБД зависит качество, способность изменения и доступное использование БД для любого человека. Главным критерием является одновременное поддержание работоспособности как и средства СУБД, так и современного языка программирования, что бывает очень сложно так как чаще всего мы используем нелицензионный продукт, который имеет свои дефекты и неисправности в ходе работы. Для реализации БД «Аэропорт» была выбрана среда разработки СУБД – MySQL, с поддержкой переноса данных на MySQLServer и поддержкой приложения для разработки MySQLWorkbench.

MySQL — свободная система управления базами данных(СУБД). MySQL является собственностью компании Oracle Corporation MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера. Гибкость MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

Основные преимущества MySQL:

* оптимизация связей с присоединением многих данных за один проход;
* записи фиксированной и переменной длины;
* гибкая поддержка форматов чисел, строк переменной длины;
* интерфейс с языками C и Web;
* быстрая работа, масштабируемость;
* совместимость с ANSI SQL;
* бесплатна в большинстве случаев;
* хорошая поддержка со стороны провайдеров услуг хостинга;
* быстрая поддержка транзакций через механизм InnoDB.

Выбор среды программирования был остановлен на языке C#, который поддерживает связь с MySQL и может работать совместно с локальным MySQLServer, совершать связь как с помощью присоединения к серверу и вызову с него определенный функций или процедур.

C# ­­– типобезопасный объектно-ориентированный язык, предназначенный для разработки разнообразных безопасных и мощных приложений, выполняемых в среде .NET Framework. С помощью языка C# можно создавать обычные приложения Windows, XML-веб-службы, распределенные компоненты, приложения "клиент-сервер", приложения баз данных и т. д. Visual C# предоставляет развитый редактор кода, конструкторы с удобным пользовательским интерфейсом, встроенный отладчик и множество других средств, упрощающих разработку приложений на базе языка C# и .NET Framework.

ВЫВОДЫ

В ходе данной курсовой работы, была разработана БД «Автовокзал», а также разработано ПО с использованием объектно-ориентированного языка программирования С# и удобной СУБД MySQL. Данная БД предназначена для персонала, управляющего всей системой автовокзалов, в нашем случае это диспетчер автовокзала, а также для любого пользователя желающего забронировать билет на выбранный рейс.

Разработанное программное обеспечение позволяет вносить изменения, добавлять или удалять данные диспетчеру по всей системе «Автовокзал», а также бронировать билеты на выбранные рейсы обычным пользователям.

Область применения данного программного обеспечения весьма широка, в частности оно может использоваться как программное средство диспетчерами автовокзалов, которые имеют доступ к данной программе.

На данный момент проект является моделью системы «Автовокзал» и может быть усовершенствован в следствии его дальнейшего развития.

ПЕРЕЧЕНЬ ССЫЛОК

1. Ульман Дж., Уидом Д. Основы систем баз данных. - М.: ЛОРИ, 2000.- 374 с.

2. http://www.infosgs.narod.ru/31.htm - Система управления базами данных (СУБД),

3. http://ru.wikipedia.org/wiki/ER-модель\_данных - ER-модель данных

4. http://www.denizzone.com/baset3r4part1.html - Физические модели БД

5. Оскерко В. С. Теория баз данных и знаний. Нормализация таблиц. – М.: БГЭУ, 2005.

6. ДСТУ 3008-95. Документация. Отчеты в сфере науки и техники. Структура и правила оформления. – Государственный стандарт Украины, 1995. – 36 с.

7. Колесник Л.В., Писклакова В.П., Писклакова О.А. Методические указания к курсовой работе по дисциплине «МОДЕЛИРОВАНИЕ ДАННЫХ И МАНИПУЛИРОВАНИЕ ДАННЫМИ». М. «Учебно-издательский издательский полиграфический центр ХНУРЭ», 2012. – 30 с

# ПРИЛОЖЕНИЕ А

Министерство образования и науки Украины

УТВЕРЖДАЮ

Руководитель курсовой работы,

доц. кафедри системотехники

\_\_\_\_\_\_\_\_\_\_\_\_ Л.В. Колесник

(подпись, дата)

БАЗА ДАННЫХ «Автовокзал»

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

ГЮИК.508100.121 – 01 12 01 – ЛУ

Студент группы \_СИ-14-2\_\_\_\_\_\_

(название группы)

Беседин Богдан Владимирович

(подпись, дата, фамилия, имя, отчество)

2016

Министерство образования и науки Украины

УТВЕРЖДАЮ

ГЮИК.508100.121 – 01 12 01 – ЛУ

БАЗА ДАННЫХ «Автовокзал»

Текст программы

ГЮИК.508100.121 – 01 12 01

Страниц \_\_28\_\_\_

2016

public class MySqlDatabase : Database

{

MySqlConnection connection;

public MySqlDatabase(string connectionQuery)

: base(connectionQuery)

{

connection = new MySqlConnection(connectionQuery);

}

public override void SendQuery(string query)

{

MySqlDataAdapter adapter = new MySqlDataAdapter();

adapter.SelectCommand = new MySqlCommand(query, connection);

connection.Open();

adapter.SelectCommand.ExecuteReader();

connection.Close();

}

public override DataTable GetData(string query)

{

DataSet dataSet = new DataSet();

MySqlDataAdapter adapter = new MySqlDataAdapter();

adapter.SelectCommand = new MySqlCommand(query, connection);

adapter.Fill(dataSet);

return dataSet.Tables[0];

}

class MySqlDatabaseBuilder:DatabaseBuilder

{

string connectionQuery = "";

public override void SetServerAddress(string address)

{

connectionQuery += "DataSource=" + address + ";";

}

public override void SetDatabaseName(string databaseName)

{

connectionQuery += "database=" + databaseName + ";";

}

public override void SetPort(int port)

{

connectionQuery += "port=" + port.ToString() + ";";

}

public override void SetUserName(string userName)

{

connectionQuery += "Username=" + userName + ";";

}

public override void SetPassword(string password)

{

connectionQuery += "Password='" + password + "';";

}

public override Database BuildDatabase()

{

return new MySqlDatabase(connectionQuery);

}

}

static class MySqlQueryConstructor

{

public static string DeleteQuery(string table, string whereQuery)

{

return "DELETE FROM " + table + " " + whereQuery;

}

public static string InsertQuery(string table, DataRow row)

{

string queryString = "INSERT INTO " + table + "(";

foreach (DataColumn column in row.Table.Columns) //table(columns)

{

queryString += column.ColumnName;

if (column != row.Table.Columns[row.Table.Columns.Count - 1]) queryString += ", ";

else queryString += ") VALUES(";

}

foreach (DataColumn column in row.Table.Columns) //value(values)

{

switch (row[column].GetType().ToString())

{

case "System.String": queryString += "'" + row[column].ToString() + "'"; break;

case "System.Int32": queryString += row[column].ToString(); break;

}

if (column != row.Table.Columns[row.Table.Columns.Count - 1]) queryString += ", ";

else queryString += ")";

}

return queryString;

}

/// <summary>

///

/// </summary>

/// <param name="order\_object">Number of column or his name</param>

/// <param name="ASC"></param>

/// <returns></returns>

public static string OrderByQuery(object order\_object, bool ASC = true)

{

string query = " ORDER BY " + order\_object;

if (ASC) query += " ASC ";

else query += " DESC ";

return query;

}

public static string SelectQuery(string table, params string[] columns)

{

string query = "SELECT ";

if (columns.Length > 0)

{

foreach (string column in columns)

{

query += column;

if (column != columns[columns.Length - 1]) query += ", ";

}

}

else query += "\*";

return query += " FROM " + table;

}

public static string SetQuery(params Dictionary<string, object>[] changes)

{

string query = "SET ";

foreach (Dictionary<string, object> change in changes)

{

foreach (string key in change.Keys)

{

query += key + " = ";

switch (change[key].GetType().ToString())

{

case "System.String": query += "'" + change[key] + "'"; break;

case "System.Int32": query += change[key].ToString(); break;

}

if (change != changes[changes.Length - 1]) query += ", ";

}

}

return query;

}

public static string UpdateQuery(string table, string setQuery)

{

return "UPDATE " + table + " " + setQuery;

}

public static string WhereQuery(params string[] conditions)

{

string query = " WHERE ";

foreach (string condition in conditions)

{

if (condition != conditions[0]) query += " AND ";

query += condition;

}

return query;

}

/// <summary>

///

/// </summary>

/// <param name="table"></param>

/// <param name="column"></param>

/// <param name="operation">such as '>', "=", "!=", ...</param>

/// <param name="value"></param>

/// <returns></returns>

public static string SimpleCondition(string column, string operation, object value)

{

string condition = column + " " + operation + " ";

switch (value.GetType().ToString())

{

case "System.String": condition += "'" + value + "'"; break;

case "System.Int32": condition += value; break;

default: condition += value.ToString(); break;

}

return condition;

}

public static string IntervalCondition(string column, object value1, object value2, bool Outside = false)

{

string condition = "";

if (!Outside) condition += column + " BETWEEN ";

else condition = column + " NOT BETWEEN ";

switch (value1.GetType().ToString())

{

case "System.String": condition += "'" + value1 + "'"; break;

case "System.Int32": condition += value1; break;

}

condition += " AND ";

switch (value2.GetType().ToString())

{

case "System.String": condition += "'" + value2 + "'"; break;

case "System.Int32": condition += value2; break;

}

return condition;

}

public static string SelectionCondition(string column, bool without, params object[] values)

{

string condition = "";

if (!without) condition = column + " IN(";

else condition = " WHERE " + column + " NOT IN(";

foreach (object value in values)

{

switch (value.GetType().ToString())

{

case "System.String": condition += "'" + value + "'"; break;

case "System.Int32": condition += value; break;

}

if (value != values[values.Length - 1]) condition += ", ";

else condition += ") ";

}

return condition;

}

public static string NULLCondition(string column, bool isNULL = true)

{

if (isNULL) return column + " IS NULL";

return column + " IS NOT NULL";

}

}

class MainViewModel : INotifyPropertyChanged

{

Page currentPage;

public MainViewModel(Page page)

{

this.currentPage = page;

EnterLikeAGuestCommand = new Command(arg => EnterLikeAGuest());

EnterLikeAnAdminCommand = new Command(arg => EnterLikeAnAdmin());

}

#region PropertyChanged

public event PropertyChangedEventHandler PropertyChanged;

void UpdatePropertyChanged(string propertyName)

{

if (PropertyChanged != null) PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

#endregion

#region Commands

public ICommand EnterLikeAGuestCommand { get; set; }

void EnterLikeAGuest()

{

currentPage.NavigationService.Navigate(new FlightsPreviewPage());

}

public ICommand EnterLikeAnAdminCommand { get; set; }

void EnterLikeAnAdmin()

{

currentPage.NavigationService.Navigate(new AdminConsolePage());

}

#endregion

}

class FlightsPreviewViewModel : INotifyPropertyChanged

{

Page currentPage;

Database database;

public FlightsPreviewViewModel(Page page)

{

this.currentPage = page;

BackToMainMenuCommand = new Command(arg => BackToMainMenu());

ApplyFiltresCommand = new Command(arg => ApplyFiltres());

ResetFiltresCommand = new Command(arg => ResetFiltres());

ReserveCommand = new Command(arg => Reserve());

CreateModel();

\_flightsTable = database.GetData(MySqlQueryConstructor.SelectQuery("Flight"));

}

void CreateModel()

{

DatabaseBuilder builder = new MySqlDatabaseBuilder();

builder.SetDatabaseName("mydb");

builder.SetServerAddress("127.0.0.1");

builder.SetPort(3306);

builder.SetUserName("root");

builder.SetPassword("");

database = builder.BuildDatabase();

}

void UpdateSelectionObservers()

{

BusInformation = "Bus:\n";

DataTable busTable = database.GetData(MySqlQueryConstructor.SelectQuery("Bus") + MySqlQueryConstructor.WhereQuery(MySqlQueryConstructor.SimpleCondition("idBus", "=", FlightsTable.Rows[SelectedIndex]["idBus"])));

foreach (DataColumn column in busTable.Columns) BusInformation += column.ColumnName + ": " + busTable.Rows[0][column] + "\n";

DriverInformation = "Driver:\n";

DataTable driverTable = database.GetData(MySqlQueryConstructor.SelectQuery("Driver") + MySqlQueryConstructor.WhereQuery(MySqlQueryConstructor.SimpleCondition("idDriver", "=", busTable.Rows[0]["idDriver"])));

foreach (DataColumn column in driverTable.Columns) DriverInformation += column.ColumnName + ": " + driverTable.Rows[0][column] + "\n";

}

#region PropertyChanged

public event PropertyChangedEventHandler PropertyChanged;

void UpdatePropertyChanged(string propertyName)

{

if (PropertyChanged != null) PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

#endregion

#region Commands

public ICommand BackToMainMenuCommand { get; set; }

void BackToMainMenu()

{

currentPage.NavigationService.Navigate(new MainPage());

}

public ICommand ReserveCommand { get; set; }

void Reserve()

{

if (SelectedIndex != -1)

{

DataRow flight = FlightsTable.Rows[SelectedIndex];

if ((int)flight["Available\_Tickets\_Amount"] > 0) currentPage.NavigationService.Navigate(new TicketReservationPage(flight));

else MessageBox.Show("К сожалению на данный рейс были проданы все билеты.", "Сожаление", MessageBoxButton.OK, MessageBoxImage.Information);

}

else MessageBox.Show("Не выбран ни один рейс!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

public ICommand ApplyFiltresCommand { get; set; }

void ApplyFiltres()

{

string query = MySqlQueryConstructor.SelectQuery("Flight");

List<string> conditions = new List<string>();

if (Departure\_Place != null && Departure\_Place != Departure\_PlacesList[0]) conditions.Add(MySqlQueryConstructor.SimpleCondition("Departure\_Place", "=", Departure\_Place));

if (Arrival\_Place != null && Arrival\_Place != Arrival\_PlacesList[0]) conditions.Add(MySqlQueryConstructor.SimpleCondition("Arrival\_Place", "=", Arrival\_Place));

if (Departure\_date != null && (Departure\_date.Year != 1)) conditions.Add(MySqlQueryConstructor.SimpleCondition(

"Departure\_date", "=", Departure\_date.Year.ToString() + "-" + Departure\_date.Month.ToString() + "-" + Departure\_date.Day.ToString()));

if (conditions.Count > 0) query += MySqlQueryConstructor.WhereQuery(conditions.ToArray());

FlightsTable = database.GetData(query);

}

public ICommand ResetFiltresCommand { get; set; }

void ResetFiltres()

{

Departure\_Place = "Place";

Arrival\_Place = "Place";

Departure\_date = new DateTime();

BusInformation = "";

DriverInformation = "";

ApplyFiltres();

}

#endregion

#region Lists

ObservableCollection<string> \_departure\_placesList;

public ObservableCollection<string> Departure\_PlacesList

{

get

{

\_departure\_placesList = new ObservableCollection<string>();

\_departure\_placesList.Add("Place");

foreach (DataRow row in FlightsTable.Rows) \_departure\_placesList.Add(row[FlightsTable.Columns["Departure\_Place"]].ToString());

return \_departure\_placesList;

}

set

{

\_departure\_placesList = value;

}

}

ObservableCollection<string> \_arriva\_placeslList;

public ObservableCollection<string> Arrival\_PlacesList

{

get

{

\_arriva\_placeslList = new ObservableCollection<string>();

\_arriva\_placeslList.Add("Place");

foreach (DataRow row in FlightsTable.Rows) \_arriva\_placeslList.Add(row[FlightsTable.Columns["Arrival\_Place"]].ToString());

return \_arriva\_placeslList;

}

set

{

\_arriva\_placeslList = value;

}

}

#endregion

#region Properties

string \_arrival\_place;

public string Arrival\_Place

{

get

{

return \_arrival\_place;

}

set

{

\_arrival\_place = value;

UpdatePropertyChanged("Arrival\_Place");

}

}

string \_departure\_place;

public string Departure\_Place

{

get

{

return \_departure\_place;

}

set

{

\_departure\_place = value;

UpdatePropertyChanged("Departure\_Place");

}

}

public DateTime Departure\_date

{

get;

set;

}

int \_selectedIndex;

public int SelectedIndex

{

get

{

return \_selectedIndex;

}

set

{

\_selectedIndex = value;

if (\_selectedIndex > -1) UpdateSelectionObservers();

}

}

string \_busInformation;

public string BusInformation

{

get

{

return \_busInformation;

}

set

{

\_busInformation = value;

UpdatePropertyChanged("BusInformation");

}

}

string \_driverInfromation;

public string DriverInformation

{

get

{

return \_driverInfromation;

}

set

{

\_driverInfromation = value;

UpdatePropertyChanged("DriverInformation");

}

}

#endregion

#region FlightTable

DataTable \_flightsTable;

public DataTable FlightsTable

{

get { return \_flightsTable; }

set

{

\_flightsTable = value;

UpdatePropertyChanged("FlightsTable");

}

}

#endregion

}

class TicketReservationViewModel : INotifyPropertyChanged

{

DataRow flight;

Database database;

Page currentPage;

public TicketReservationViewModel(Page page, DataRow flight)

{

this.currentPage = page;

this.flight = flight;

BackToFlightPreviewCommand = new Command(arg => BackToFlightPreview());

ApplyReservationCommand = new Command(arg => ApplyReservation());

CreateModel();

}

void CreateModel()

{

DatabaseBuilder builder = new MySqlDatabaseBuilder();

builder.SetDatabaseName("mydb");

builder.SetServerAddress("127.0.0.1");

builder.SetPort(3306);

builder.SetUserName("root");

builder.SetPassword("");

database = builder.BuildDatabase();

}

#region PropertyChanged

public event PropertyChangedEventHandler PropertyChanged;

void UpdatePropertyChanged(string propertyName)

{

if (PropertyChanged != null) PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

#endregion

#region Commands

public ICommand BackToFlightPreviewCommand { get; set; }

void BackToFlightPreview()

{

currentPage.NavigationService.Navigate(new FlightsPreviewPage());

}

public ICommand ApplyReservationCommand { get; set; }

void ApplyReservation()

{

if ((PassengerSurname != null && PassengerSurname != "") && (PassengerName != null && PassengerName != "") && (PassengerPatronymic != null && PassengerPatronymic != "") && (PassengerNationality != null && PassengerNationality != ""))

{

ReserveTicket();

BackToFlightPreview();

}

else MessageBox.Show("Все поля должны быть заполнены!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

void ReserveTicket()

{

DataTable passengerTable = new DataTable();

foreach (PropertyInfo property in typeof(Passenger).GetProperties()) if (property.Name != "idPassenger") passengerTable.Columns.Add(property.Name);

DataRow passenger = passengerTable.NewRow();

passenger["Surname"] = PassengerSurname;

passenger["Name"] = PassengerName;

passenger["Patronymic"] = PassengerPatronymic;

passenger["Nationality"] = PassengerNationality;

database.SendQuery(MySqlQueryConstructor.InsertQuery("Passenger", passenger));

Dictionary<string, object> changes = new Dictionary<string, object>();

changes.Add("Available\_Tickets\_Amount", (int)flight["Available\_Tickets\_Amount"] - 1);

database.SendQuery(MySqlQueryConstructor.UpdateQuery("Flight", MySqlQueryConstructor.SetQuery(changes)) +

MySqlQueryConstructor.WhereQuery(MySqlQueryConstructor.SimpleCondition("idFlight", "=", flight["idFlight"])));

}

#endregion

#region Properties

public string PassengerSurname

{

get;

set;

}

public string PassengerName

{

get;

set;

}

public string PassengerPatronymic

{

get;

set;

}

public string PassengerNationality

{

get;

set;

}

#endregion

}

class AdminConsoleViewModel : INotifyPropertyChanged

{

Database database;

Page currentPage;

public AdminConsoleViewModel(Page page)

{

this.currentPage = page;

BackToMainMenuCommand = new Command(arg => BackToMainMenu());

CreateModel();

}

void CreateModel()

{

DatabaseBuilder builder = new MySqlDatabaseBuilder();

builder.SetDatabaseName("mydb");

builder.SetServerAddress("127.0.0.1");

builder.SetPort(3306);

builder.SetUserName("root");

builder.SetPassword("");

database = builder.BuildDatabase();

}

#region PropertyChanged

public event PropertyChangedEventHandler PropertyChanged;

void UpdatePropertyChanged(string propertyName)

{

if (PropertyChanged != null) PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

#endregion

#region Commands

public ICommand BackToMainMenuCommand { get; set; }

public void BackToMainMenu()

{

currentPage.NavigationService.Navigate(new MainPage());

}

#endregion

#region Properties

TabItem \_selectedTab;

public TabItem SelectedTab

{

get

{

return \_selectedTab;

}

set

{

\_selectedTab = value;

if (\_selectedTab.Name == "Flights") CurrentTable = database.GetData(MySqlQueryConstructor.SelectQuery("Flight"));

if (\_selectedTab.Name == "Passengers") CurrentTable = database.GetData(MySqlQueryConstructor.SelectQuery("Passenger"));

}

}

DataTable \_currentTable;

public DataTable CurrentTable

{

get

{

return \_currentTable;

}

set

{

\_currentTable = value;

UpdatePropertyChanged("CurrentTable");

}

}

#endregion

}

# ПРИЛОЖЕНИЕ Б

Министерство образования и науки Украины

УТВЕРЖДАЮ

Руководитель курсовой работы,

доц. кафедры системотехники

\_\_\_\_\_\_\_\_\_\_\_\_ Л.В. Колесник

(подпись, дата)

БАЗА ДАННЫХ «АВТОВОКЗАЛ»

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

ГЮИК.508100.121 – 01 12 01 – ЛУ

Студент группы \_СИ-14-2\_\_\_\_\_\_

(название группы)

Беседин Богдан Владимирович

(подпись, дата, фамилия, имя, отчество)

2016

Министерство образования и науки Украины

УТВЕРЖДАЮ

ГЮИК.508100.121 ИЗ – ЛУ

БАЗА ДАННЫХ «АВТОВОКЗАЛ»

Руководство пользователя

ГЮИК.508100.121 ИЗ

Страниц \_8\_

2016ВВЕДЕНИЕ

Данная база данных разработана с учетом особенностей предметной области. Целью данного курсового проекта является разработка базы данных для работы автовокзала. Любой пользователь, даже не имея опыта работы с базой данных, сможет легко работать с ней, добавляя, обновляя и удаляя информацию базы данных.

Б.1 НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ

Разработанная база данных «Автовокзал» предназначена для использования любым пользователем. Данный программный продукт создаётся для систематизации данных о рейсах, пассажиров, автобусов, персонала и билетов. Для использования данного программного продукта необходимо наличие компьютера с минимальными необходимыми системными требованиями, указанными в следующем разделе, установленный локальный сервер.

Б.2 ПОДГОТОВКА К РАБОТЕ

Необходимые ресурсы для работы:

* процесcор Intel Celeron E3300;
* минимальный объем оперативной памяти - 256 Мбайт;
* жесткий диск с объемом свободного пространства не менее 10 Гбайт;
* рекомендуется монитор типа VGA или лучшего разрешения;
* CD-ROM, клавиатура, мышь.

Для запуска данной базы данных необходимо иметь компьютер с операционной системой Windows или Linux.

Б.3 ОПИСАНИЕ ОПЕРАЦИЙ

Когда мы запускаем программу, мы первым делом видим главное меню, представленное на рисунке Б.3.1, на котором расположена специальная форма, для авторизации администратора, и кнопка «Вход» для входа в режиме администратора. Также любой желающий может зайти без авторизации, просто нажав кнопку «Войти как гость».

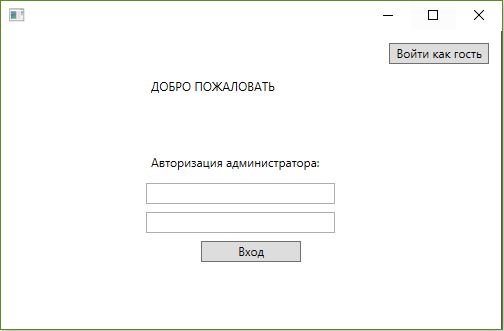


Рисунок Б.3.1 – Главное меню

Путем нажатия кнопки «Войти как гость» мы оказываемся в режиме гостя, и переходим в меню просмотра рейсов изображенное на рисунке Б.3.2. Либо после авторизации пользователь может зайти в режиме администратора в окно редактирования таблиц изображенное на рисунке Б 3.4.

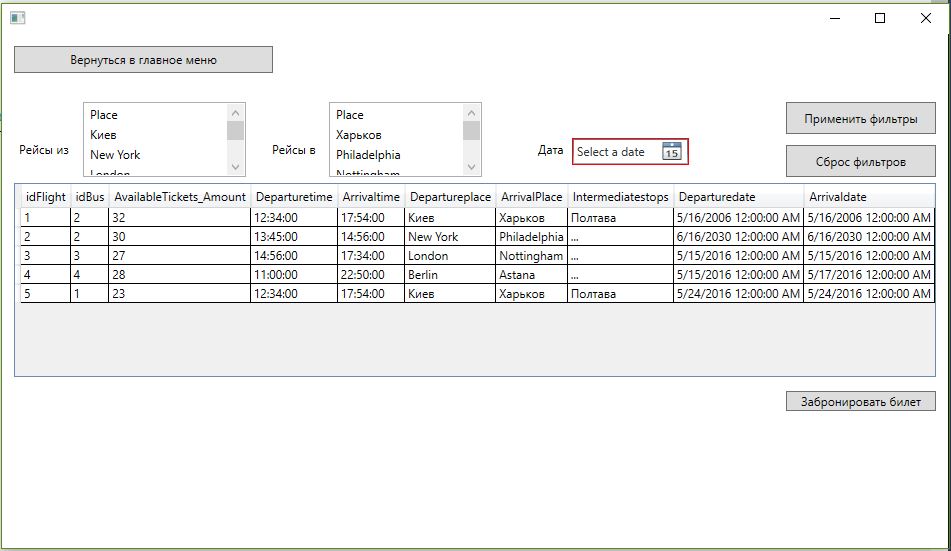


Рисунок Б.3.2 – Меню рейсов программы «Автовокзал»

В данном меню пользователь может выбирать подходящий ему рейс из списка имеющихся. Также пользователь может воспользоваться набором фильтров, таких как: «Рейсы из» - выбор места отправки автобуса, «Рейсы в» - выбор места прибытия автобуса», «Дата» - дата прибытия автобуса, и нажать кнопку «Применить фильтры» для фильтрации содержимого таблицы по заданным критериям, затем пользователь может нажать кнопку «Сброс фильтров» для отображения всех имеющихся рейсов без применения к ним фильтров. Также после выбора нужного рейса пользователь может нажать кнопку «Забронировать билет» для перехода в окно бронирования билетов, изображенного на рисунке Б 3.3.

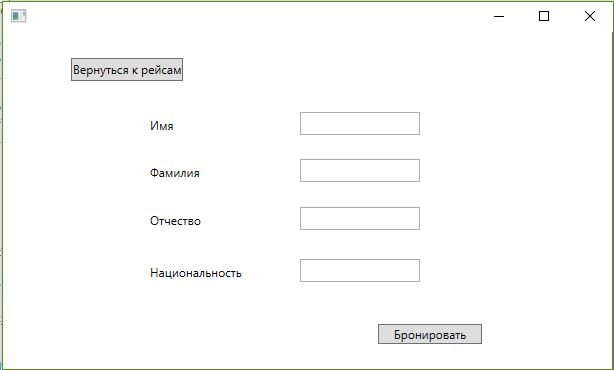


Рисунок Б.3.3 – Меню бронирования билетов

В меню бронирования билетов пользователю предоставляется специальная форма, которую он обязан заполнить, для корректного бронирования билета.

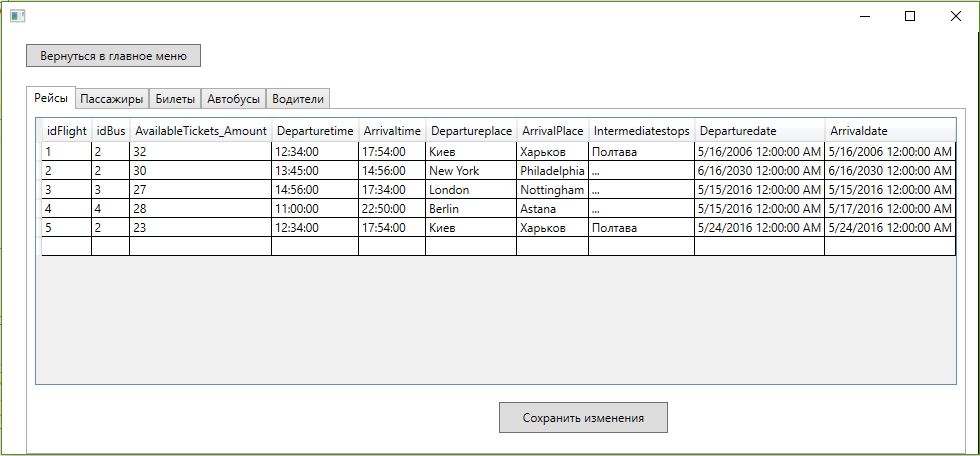


Рисунок Б.3.4 – «Окно редактирования таблиц»

В данном окне администратор имеет право вносить изменения в имеющиеся таблицы, и нажать кнопку «Сохранить изменения» для сохранения проделанной работы в базе данных.

Б.4 РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ

При освоении данной программы у пользователя не должно возникнуть проблем, для этого необходимо внимательно прочитать «Руководство пользователя» и выполнять все действия в соответствии с заданными требованиями.