МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

|  |  |
| --- | --- |
|  | Кафедра вычислительной техники |

**РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА**

**ПО ДИСЦИПЛИНЕ «ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

**на тему: «Информационная система “Деятельность авторемонтной мастерской”»**

|  |
| --- |
|  |

|  |
| --- |
| Студент: Дунаев Н. Ю. |

|  |
| --- |
| Группа: АВТ-610 |

|  |
| --- |
| Преподаватель: Бычков М.И. |

|  |
| --- |
|  |

Новосибирск 2019 г.

Оглавление

[1. Задание 3](#_Toc9976675)

[2. Формирование требований к базе данных информационной системы 4](#_Toc9976676)

[3. Структура информационной системы 5](#_Toc9976677)

[4. Структура Back-End 5](#_Toc9976678)

[5. Структура Front-End 7](#_Toc9976679)

[6. Среда разработки 7](#_Toc9976680)

[7. Back-End реализация 8](#_Toc9976681)

[8. Front-End реализация 9](#_Toc9976682)

[9. Описание алгоритмов обмена данными с БД 15](#_Toc9976683)

[10. Выводы 18](#_Toc9976684)

# Задание

Разработать информационную систему, которая автоматизирует деятельность магазина.

**Система должна:**

1. содержать сведения по рабочим мастерской (имя, фамилия, номер, специальность (кузовщик, электрик, слесарь и т.д.), дата приема на работу;
2. сведения о находящихся в ремонте машинах (тип, регистрационный номер, дату поступления в ремонт, характер неисправности, порядковый номер, присвоенный в мастерской, дату окончания ремонта (если он завершен), стоимость завершенного ремонта;
3. сведения о ремонте: порядковый номер, тип ремонта, список рабочих, ремонтирующих машину;
4. выдавать справки о доходах мастерской за месяц, год;
5. справки о машинах, находящихся в ремонте и по типу ремонта;
6. список рабочих, участвующих в ремонте данной машины;
7. иметь средства ввода и обновления всех данных.

# Формирование требований к базе данных информационной системы

В проектируемой ИС должна содержаться следующая информация:

* Данные о сотрудниках (таблица stuff)
* Данные об автомобилях (таблица wheels)
* Данные о ремонте (таблица repair)

В планируемых таблицах соответственно должны быть следующие поля:

* Таблица “Сотрудники” (stuff)
  + Идентификатор (id)
  + Имя (name)
  + Фамилия (surname)
  + Дата начала работы (employ\_date)
  + Специализация (speciality)
* Таблица “Автомобили” (wheels)
  + Идентификатор (id)
  + Ремонтный номер (repair\_number)
  + Тип (type)
  + Модель (model)
  + Регистрационный номер (reg\_number)
  + Дата поступления (arrival\_date)
  + Неисправность (defect)
  + Дата окончания ремонта (end\_date)
  + Цена ремонта (price)
* Таблица “Ремонт” (repair)
  + Идентификатор (id)
  + Тип ремонта (type)
  + Список рабочих (workers)

# Структура информационной системы



Рисунок 1. Структура информационной системы (IDEF0)

Решение поставленной задачи предполагает декомпозицию проекта на два компонента: Front-End и Back-End.

Back-End в нашем случае будет представлен базой данных мастерской, которая будет содержать данные о её работниках, обслуживаемых автомобилях и непосредственно сервисных проектах, а также осуществлять обработку запросов.

Front-End частью является пользовательское приложение, связывающее базу данных и её администратора (пользователя системы). Приложение предоставляет простой для изучения и использования интерфейс, реализует функции, необходимые для управления системой и в то же время скрывает прямой доступ к данным, в целях обеспечения безопасности.

# Структура Back-End

Ниже представлены описания таблиц с указанием типов данных столбцов.

Таблица 1. Сущность «Автомобили» (wheels)

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| Идентификатор (id) | Serial PK |
| Ремонтный номер (repair\_number) | integer |
| Тип (type) | Varchar(40) |
| Модель (model) | Varchar(40) |
| Регистрационный номер (reg\_number) | Varchar(40) |
| Дата поступления (arrival\_date) | date |
| Неисправность (defect) | Varchar(40) |
| Цена ремонта (price) | integer |

Таблица 2. Сущность «Сотрудники» (stuff)

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| Идентификатор (id) | Serial PK |
| Имя (name) | Varchar(40) |
| Фамилия (surname) | Varchar(40) |
| Дата начала работы (employ\_date) | date |
| Специализация (speciality) | Varchar(40) |

Таблица 3. Сущность «Ремонт» (repair)

|  |  |
| --- | --- |
| Название столбца | Тип данных |
| Идентификатор (id) | Serial PK |
| Тип ремонта (type) | Varchar(40) |
| Список рабочих (workers) | Varchar(40) |

Представленные сущности имеют связи, описанные следующей диаграммой:

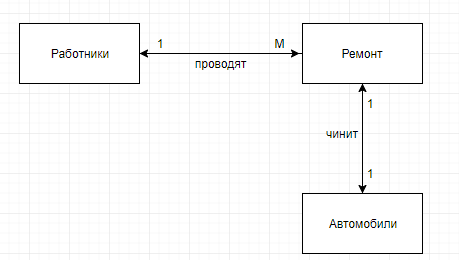


Рисунок 2. Диаграмма связей сущностей базы данных.

# Структура Front-End

Ниже представлена диаграмма графических форм, отображающая связи окон приложения.



Рисунок 3. Диаграмма форм графического интерфейса приложения.

# Среда разработки

Для разработки информационной системы были выбраны следующие инструменты:

1. СУБД PostgreSQL. Данная СУБД позволяет хранить все данные информационной системы в базе данных на сервере. Таким образом, все клиенты системы будут обращаться к одной базе данных посредством ИС.

К преимуществам данной СУБД относят:

* быстродействие;
* высокий уровень безопасности;
* надежность;
* переносимость.

1. Среда программирования Microsoft Visual Studio 2013.
2. Объектно-ориентированный язык программирования C#. Данный язык хорошо подходит для создания множества приложений, работающих в среде .NET Framework. Благодаря множеству нововведений C# обеспечивает возможность быстрой и лёгкой разработки desktop-приложений.

Visual C# — это реализация языка C# корпорацией Майкрософт. Поддержка Visual C# в Visual Studio обеспечивается с помощью полнофункционального редактора кода, компилятора, шаблонов проектов, конструкторов, мастеров кода, мощного и удобного отладчика и многих других средств. Библиотека классов .NET Framework предоставляет доступ ко многим службам операционной системы и к другим полезным, хорошо спроектированным классам, что существенно ускоряет цикл разработки.

# Back-End реализация

CREATE DATABASE repair;

CREATE TABLE stuff(

id SERIAL PRIMARY KEY,

Name varchar(40),

Surname varchar(40),

Speciality varchar(40) CHECK (Speciality LIKE 'Body worker' OR Speciality LIKE 'Electric' OR Speciality LIKE 'Mechanic'),

Employ\_date date CHECK(Employ\_date > '1900-01-01')

);

CREATE TABLE wheels(

id SERIAL PRIMARY KEY,

Repair\_number int,

Type varchar(40) CHECK(Type LIKE 'Legkovoi' OR Type LIKE 'Gryzovoi'),

Model varchar(40),

Reg\_number varchar(15),

Arrival\_date date CHECK(Arrival\_date > '1900-01-01'),

Defect varchar(40),

End\_date date,

Price int

);

CREATE TABLE repair(

id SERIAL PRIMARY KEY,

Type varchar(40),

Workers varchar(40)

);

ALTER TABLE wheels add check (Arrival\_date < End\_date);

# Front-End реализация

Для функционирования приложения для взаимодействия с базой данных PostgreSQL в проект был подключён драйвер Npgsql.

Для создания нового подключения к базе данных нужно сформировать строку подключения. Она выглядит следующим образом:

*NpgsqlConnection("server=localhost;database=repair;UserId=postgres;password=12345");*

Далее для работы с базой данных используются такие компоненты, как Адаптеры, команды и контейнеры данных DataSet.

Интерфейс приложения состоит из окон, представленных 9 разными формами.

* Главная форма



Рисунок 4. Главная форма

Чтобы начать работу, необходимо сначала загрузить базу. Открываем меню(Файл) и выбираем пункт «Загрузить» или нажимаем горячую комбинацию клавиш Ctrl+O:

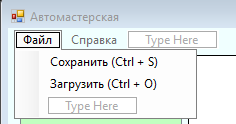


Рисунок 5. Меню (Файл).

При нажатии на пункт «Загрузить» пользователь может просматривать список таблиц.

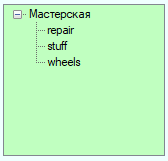


Рисунок 6. Список таблиц.

Пользователь может дважды кликнуть на любую таблицу и просмотреть ее содержимое. Вместе с этим, станет доступна возможность редактирования таблиц (добавление новой записи, редактирование или удаление существующих записей). При наведении на кнопку «Добавить», «Удалить» или «Изменить» появляется соответствующая подсказка.

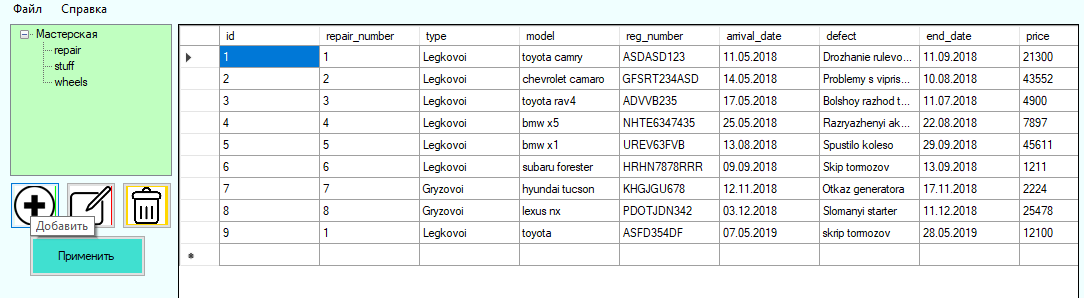


Рисунок 7. Просмотр таблиц.

В зависимости от выбранной таблицы, пользователь, нажав на кнопку «Добавить», «Удалить» или «Изменить» попадает в соответствующее окно:

* Добавить автомобиль (Выбрана таблица «Автомобили», нажата кнопка «Добавить»

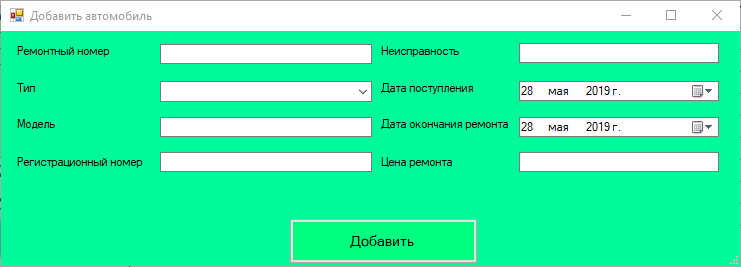


Рисунок 8. Добавить автомобиль.

* Удалить автомобиль (Выбрана таблица «Автомобили», нажата кнопка «Удалить»

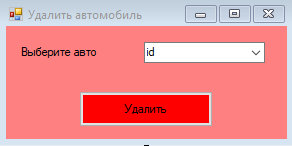


Рисунок 9. Удалить автомобиль.

* Изменить автомобиль (Выбрана таблица «Автомобили», нажата кнопка «Изменить»

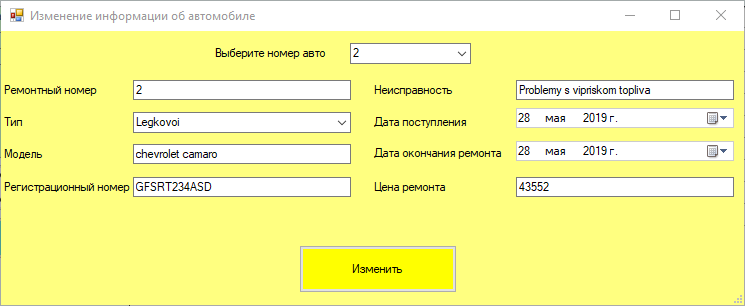


Рисунок 10. Изменить автомобиль.

* Добавить сотрудника (Выбрана таблица «Сотрудники», нажата кнопка «Добавить»



Рисунок 11. Добавить сотрудника.

* Удалить сотрудника (Выбрана таблица «Сотрудники», нажата кнопка «Удалить»

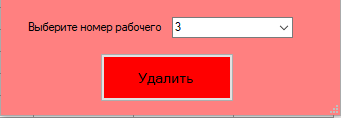


Рисунок 12. Удалить сотрудника.

* Изменить сотрудника (Выбрана таблица «Сотрудники», нажата кнопка «Изменить»

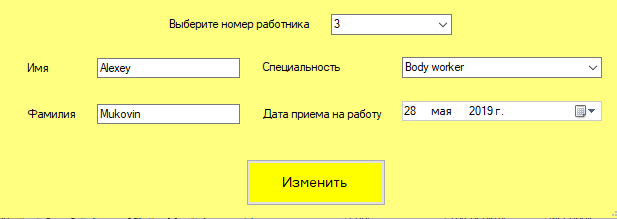


Рисунок 13. Изменить сотрудника.

* Добавить ремонт (Выбрана таблица «ремонт», нажата кнопка «Добавить»

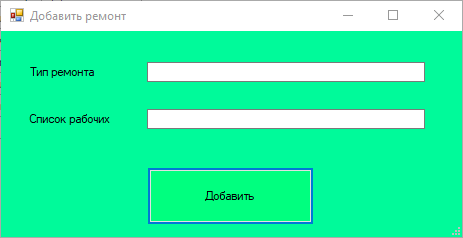


Рисунок 14. Добавить ремонт.

* Удалить ремонт (Выбрана таблица «ремонт», нажата кнопка «Удалить»

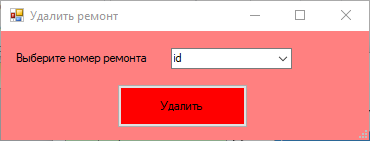


Рисунок 15. Удалить ремонт.

* Изменить ремонт (Выбрана таблица «ремонт», нажата кнопка «Изменить»

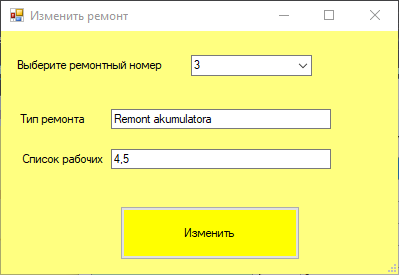


Рисунок 16. Изменить ремонт.

* Справка о доходах

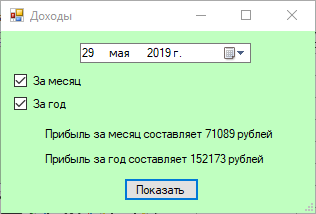


Рисунок 17. Справка о доходах.

* Справка об автомобилях

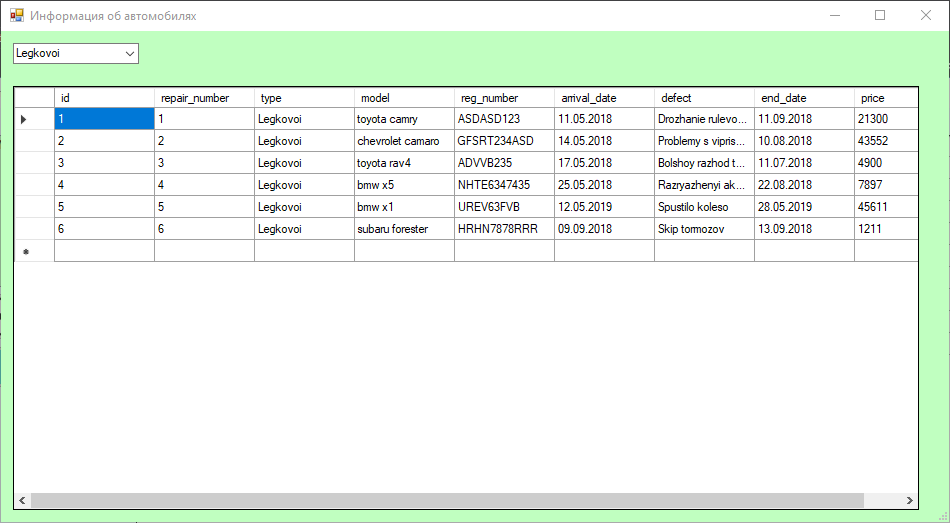


Рисунок 18. Справка об автомобилях.

* Справка о работниках

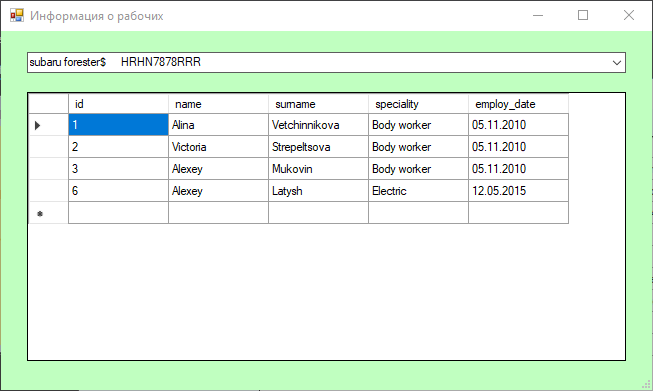


Рисунок 19. Справка о работниках.

После редактирования таблиц пользователь может нажать кнопку «Применить, чтобы сохранить данные в БД.

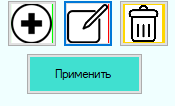


Рисунок 20. Кнопка «Применить».

# Описание алгоритмов обмена данными с БД

При нажатии на кнопку «Загрузить» формируется запрос информации о таблицах БД.

string query = "SELECT table\_name FROM information\_schema.tables WHERE table\_schema NOT IN ('information\_schema','pg\_catalog');";

NpgsqlDataAdapter adapterTables = new NpgsqlDataAdapter(query, conn);

DataSet dataSet1 = new DataSet();

TreeNode databaseNode = new TreeNode("Databases");

adapterTables.Fill(dataSet1, "table\_name");

DataTable table1;

table1 = dataSet1.Tables["table\_name"];

TreeNode databaseNode1 = new TreeNode("Мастерская");

foreach (DataRow current1 in table1.Rows)

{

TreeNode databaseNode2 = new TreeNode(current1["table\_name"].ToString());

databaseNode1.Nodes.Add(databaseNode2);

}

databaseNode.Nodes.Add(databaseNode1);

treeView1.Nodes.Add(databaseNode1);

Редактирование таблиц происходит в оперативной памяти.

// Добавление нового автомобиля

if(textBox1.Text == "" || comboBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "" || textBox5.Text == "")

{

label9.Visible = true;

}

else

{

mainForm.dset.Tables[0].Rows.Add(mainForm.dset.Tables[0].Rows.Count + 1, textBox1.Text, comboBox1.Text, textBox2.Text, textBox3.Text, dateTimePicker1.Value.ToString("dd.MM.yyyy"), textBox4.Text, dateTimePicker2.Value.ToString("dd.MM.yyyy"), textBox5.Text);

mainForm.dset.AcceptChanges();

this.Close();

}

// Изменение автомобиля

if (textBox1.Text == "" || comboBox2.Text == "" || textBox2.Text == "" || textBox3.Text == "" || textBox4.Text == "" || textBox5.Text == "")

{

label10.Visible = true;

}

else

{

mainForm.dataGridView1.Rows[comboBox1.SelectedIndex].Cells[1].Value = textBox1.Text;

mainForm.dataGridView1.Rows[comboBox1.SelectedIndex].Cells[2].Value = comboBox2.Text;

mainForm.dataGridView1.Rows[comboBox1.SelectedIndex].Cells[3].Value = textBox2.Text;

mainForm.dataGridView1.Rows[comboBox1.SelectedIndex].Cells[4].Value = textBox3.Text;

mainForm.dataGridView1.Rows[comboBox1.SelectedIndex].Cells[6].Value = textBox4.Text;

mainForm.dataGridView1.Rows[comboBox1.SelectedIndex].Cells[8].Value = textBox5.Text;

mainForm.dataGridView1.Rows[comboBox1.SelectedIndex].Cells[5].Value = dateTimePicker1.Value.ToString("dd-MM-yyyy");

mainForm.dataGridView1.Rows[comboBox1.SelectedIndex].Cells[7].Value = dateTimePicker2.Value.ToString("dd-MM-yyyy");

this.Close();

}

// Удаление автомобиля

for (int i = 0; i < mainForm.dset.Tables[0].Rows.Count; i++)

{

if (Convert.ToInt32(mainForm.dataGridView1.Rows[i].Cells[0].Value) == Convert.ToInt32(comboBox1.SelectedItem))

{

mainForm.dset.Tables[0].Rows[i].Delete();

}

}

mainForm.dset.AcceptChanges();

// Загрузка возможных автомобилей в выпадающий список

for (int i = 0; i < mainForm.dataGridView1.RowCount - 1; i++)

{

comboBox1.Items.Add(mainForm.dataGridView1.Rows[i].Cells[0].Value);

}

При нажатии на кнопку «Принять» все измененные данные отправляются в БД.

// Отправка даных в БД на примере таблицы автомобилей

string clear = "DELETE FROM wheels;";

string newSeq = "ALTER SEQUENCE wheels\_id\_seq RESTART WITH 1;";

string req = "INSERT INTO wheels(Repair\_number, Type, Model, Reg\_number, Arrival\_date, Defect, End\_date, Price) VALUES ";

dataGridView1.Columns[5].DefaultCellStyle.Format = "yyyy-MM-dd";

dataGridView1.Columns[7].DefaultCellStyle.Format = "yyyy-MM-dd";

int i;

for(i = 0; i < dset.Tables[0].Rows.Count - 1; i++)

{

req += "(";

req = req + dataGridView1.Rows[i].Cells[1].Value + ", '" + dataGridView1.Rows[i].Cells[2].Value + "', '" + dataGridView1.Rows[i].Cells[3].Value + "', '" + dataGridView1.Rows[i].Cells[4].Value + "', '" + dataGridView1.Rows[i].Cells[5].FormattedValue + "', '" + dataGridView1.Rows[i].Cells[6].Value + "', '" + dataGridView1.Rows[i].Cells[7].FormattedValue + "', " + dataGridView1.Rows[i].Cells[8].Value;

req += "),";

}

req += "(";

req = req + dataGridView1.Rows[i].Cells[1].Value + ", '" + dataGridView1.Rows[i].Cells[2].Value + "', '" + dataGridView1.Rows[i].Cells[3].Value + "', '" + dataGridView1.Rows[i].Cells[4].Value + "', '" + dataGridView1.Rows[i].Cells[5].FormattedValue + "', '" + dataGridView1.Rows[i].Cells[6].Value + "', '" + dataGridView1.Rows[i].Cells[7].FormattedValue + "', " + dataGridView1.Rows[i].Cells[8].Value;

req += ");";

Console.WriteLine(req);

NpgsqlCommand command;

conn.Open();

command = new NpgsqlCommand(clear, conn);

command.ExecuteScalar();

command = new NpgsqlCommand(newSeq, conn);

command.ExecuteScalar();

command = new NpgsqlCommand(req, conn);

command.ExecuteNonQuery();

conn.Close();

NpgsqlDataAdapter adap = new NpgsqlDataAdapter("SELECT \* FROM " + selectedTable + ";", conn);

dset = new DataSet();

adap.Fill(dset);

dataGridView1.AutoGenerateColumns = true;

dataGridView1.DataSource = dset;

dataGridView1.DataMember = "table";

# 10. Выводы

При выполнении расчетно-графической работы была разработана информационная система, которая автоматизирует работу ремонтной автомастерской. Был получен ряд полезных навыков, таких как планирование и реализация сложной информационной системы, используя несколько программных компонентов. Получены новые знания о том, как автоматически загружать/выгружать базу данных, не обращаясь при этом к пользователю. Получены умения проектирования понятного и приятного дизайна информационной системы. Планирование информационной системы – процесс важный и требующий особого внимания, так как без четкого понимания цели проектирования, выделения ключевых моментов требования к ИС, ее реализация по большому счету не имеет особого смысла.