|  |  |
| --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  РОССИЙСКОЙ ФЕДЕРАЦИИ | |
| Федеральное государственное бюджетное образовательное  учреждение высшего образования «Пермский государственный национальный  исследовательский университет» | |
|  | Кафедра математического обеспечения вычислительных систем |
| Отчет по дисциплине: Учебная практика по базам данных и СУБД | |
| “Автоматизированное рабочее место” | |
|  | Выполнила:  студентка группы ФИТ 2 курса механико-математического факультета Фролова Екатерина |
|  | Принял:  ассистент кафедры «МОВС» Постаногов Игорь Сергеевич |
| Пермь 2017 | |

# Постановка задачи

Написать программу, которая на графическом интерфейсе пользователя позволяет просматривать, добавлять, удалять, изменять записи в 4 связанных таблицах из спроектированной ранее базы данных предметной области без необходимости задания и отображения пользователю искусственных ключей.

При запуске программы пользователю должна отображаться форма аутентификации по логину и паролю, допускающая гостевой вход. Из формы аутентификации должна быть возможность перейти на форму регистрации нового логина и пароля для оператора АРМ.

Пользователь, зашедший в приложение под учётной записью администратора АРМ, должен иметь возможность просматривать, добавлять, удалять, изменять записи во всех таблицах из спроектированной ранее базы данных предметной области и в таблице учётных записей пользователей. Пользователь, зашедший в приложение под учётной записью оператора АРМ, должен иметь возможность просматривать, добавлять, удалять, изменять записи только во всех таблицах из спроектированной ранее базы данных предметной области. Пользователь, зашедший в приложение под гостевой учётной записью АРМ, должен иметь возможность только просматривать записи во всех таблицах из спроектированной ранее базы данных предметной области.

Провести тестирование корректности работы программы.

# Анализ предметной области

Для решения задачи была выбрана предметная область «Клиника», в которой были выделены следующие сущности: пациент, врач, запись на прием, прививки; вспомогательная сущность – вакцины (тут нас будет интересовать только название вакцины и ее идентификатор)

У пациента есть личный номер (id), ФИО, дата рождения, номер телефона и пол.

Врач также обладает личным номером (id), именем, номером телефона, специализацией и возрастными ограничениями (взрослый, детский или общий).

При записи на прием необходимо указать какой пациент пойдет на этот прием (id пациента), какой врач его (прием) проведет (id врача), номер кабинета, дату, время и примечание (необязательное поле). У каждого врача может быть несколько пациентов и каждый пациент может обратиться к нескольким врачам.

Про каждую прививку известно: идентификатор, кому была поставлена прививка (id пациента), id вакцины и дата. Любая прививка относится только к одному пациенту, но у каждого пациента может быть несколько прививок.

Исходя из этого, в базе данных созданы основные таблицы «Пациенты», «Врачи», «Запись» и «Прививки». Структуры таблицы представлены в табл. 1 - 4. Вспомогательная таблица «Вакцины», структуры которой представлена в табл. 5.

Таблица 1. Структура таблицы «Пациенты»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Разрешить NULL |
| **Id пациента** | int | Нет |
| ФИО пациента | nvarchar(50) | Нет |
| Пол | nchar(11) | Нет |
| Дата рождения | date | Нет |
| Телефон | nchar(11) | Нет |

Таблица 2. Структура таблицы «Врачи»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Разрешить NULL |
| **Id врача** | int | Нет |
| ФИО врача | nvarchar(50) | Нет |
| Телефон | nchar(11) | Нет |
| специализация | nvarchar(50) | Нет |
| Возрастные ограничения | nvarchar(50) | Нет |

Таблица 3. Структура таблицы «Запись»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Разрешить NULL |
| **Id записи** | int | Нет |
| Id врача | int | Нет |
| Id пациента | Int | Нет |
| Номер кабинета | Int | Нет |
| Дата приема | date | Нет |
| Время приема | time(7) | Нет |
| Примечание | nvarchar(MAX) | Да |

Таблица 4. Структура таблицы «Прививки»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Разрешить NULL |
| **Id прививки** | Int | Нет |
| Id пациента | Int | Нет |
| Id вакцины | Int | Нет |
| Дата | Datetime | Нет |

Таблица 5. Структура таблицы «Вакцины»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Разрешить NULL |
| **Id вакцины** | Int | Нет |
| Название | nvarchar(50) | Нет |

Для хранения зарегистрированных пользователей используется таблица «Users», в которой хранится логин, зашифрованный пароль, соль, статус пользователя (администратор или оператор) и дата регистрации.

Логин уникален. Пароль сохраняется в виде хеш-кода сгенерированного для последовательности «соль»+«пароль».

В таблице 6 представлены выбранные типы данных

Таблица 6. Структура таблицы «Users»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Разрешить NULL |
| **Id пользователя** | Int | нет |
| Login | nvarchar(50) | нет |
| Salt | char(44) | нет |
| Password | char(64) | нет |
| Статус | nvarchar(20) | нет |
| Date | Datetime | нет |

# Описание функционала

Внешний вид и функционал приложения

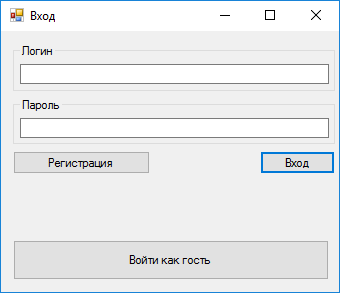
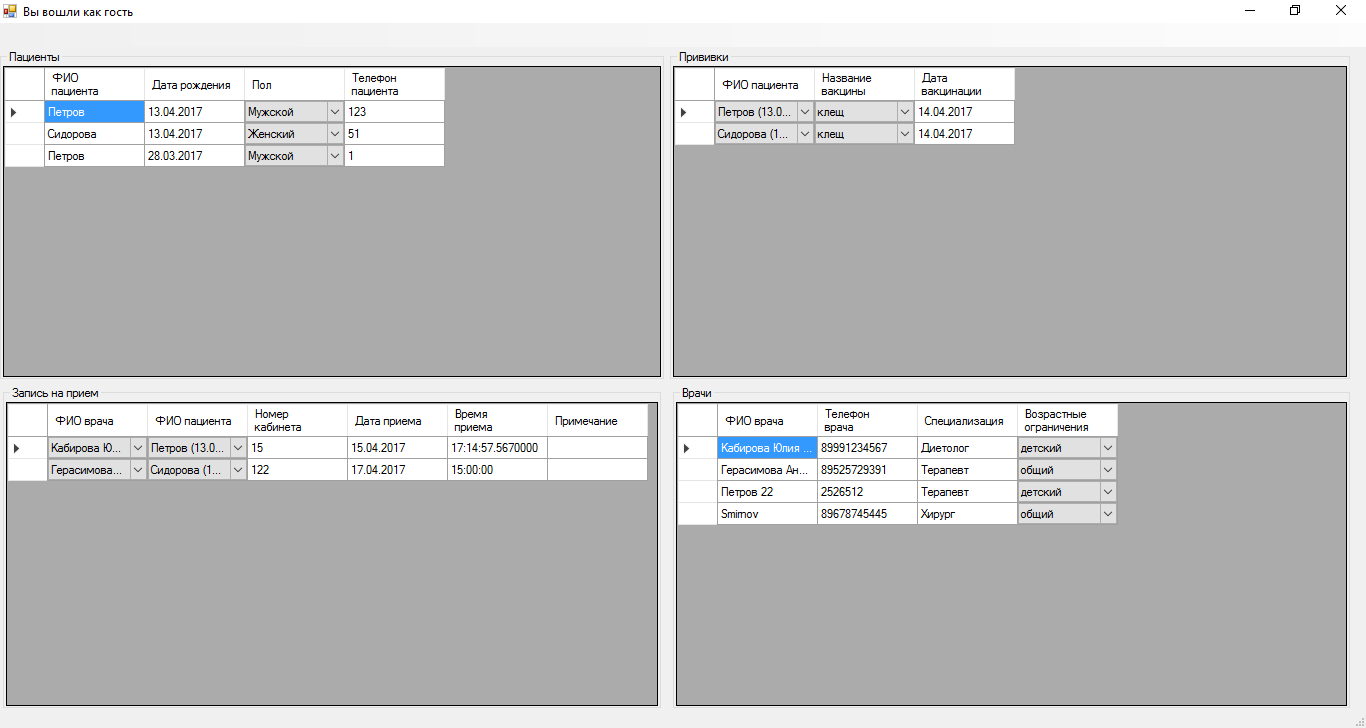
При запуске приложения отображается форма с двумя полями типа textbox для ввода логина и пароля, кнопкой «Вход», кнопкой «Войти как гость» и кнопкой «Регистрация» (рис. 1).

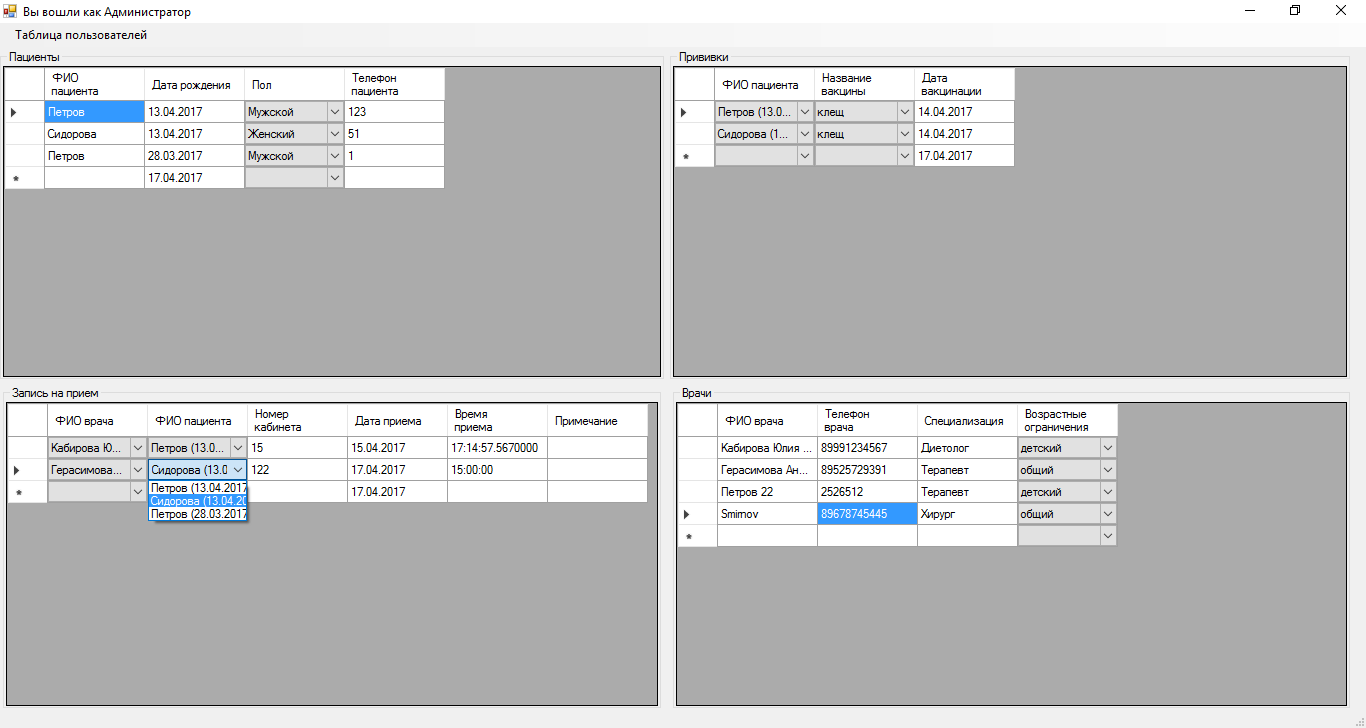
Рис 1. Внешний вид формы входа

При нажатии на кнопку «Регистрация» происходит регистрация нового пользователя с данными из textbox (при условии, что данные удовлетворяют всем условиям, в противном случае будет выдано соответствующее уведомление).

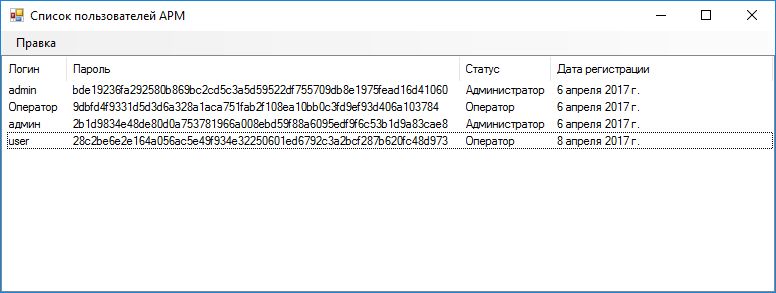
При нажатии на кнопку «Войти как гость» открывается главная форма, на которой расположены 4 элемента управления DataGridView (рис. 2). Возможность добавлять, изменять или удалять записи отсутствует.

При нажатии на кнопку «Войти» происходит проверка существует ли пользователь с данными введенными в textbox. Если такой пользователь не найден, выводится соответствующее сообщение. Если такой пользователь существует открывается главная форма (рис 3). Есть возможность добавлять, изменять или удалять записи. Заголовок главной формы информирует пользователя о статусе его учетной записи.

Рис 2. Внешний вид главной формы для входа без учетной записи

Рис 3. Внешний вид главной формы для входа через учетную запись

Если пользователь является администратором АРМ, то на главной форме доступен пункт меню «Таблица пользователей», при нажатии на которую открывается форма с данными о зарегистрированных пользователях (рис. 4).

Рис 4. Внешний вид формы

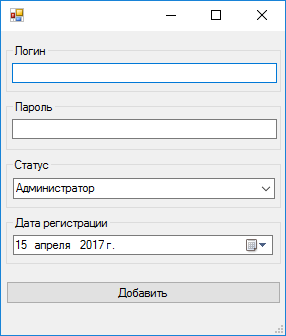
На форме расположен элемент ListView для отображения данных и кнопка меню «Правка», содержащая подменю «Добавить», «Изменить» и «Удалить». При нажатии на кнопку «Добавить» или «Изменить» отображается форма добавления/изменения (рис. 5), на которой расположены два элемента textbox для ввода логина и пароля, элемент DateTimePicker для выбора даты регистрации, combobox для выбора статуса будущего пользователя, кнопка «Добавить»/«Изменить», текст на которой меняется в зависимости от того, добавляется ли или изменяется учетная запись.

Рис 5. Внешний вид формы добавления/изменения при добавлении записи

При изменении учетной записи в форму добавления/изменения выгружаются имеющиеся значения.

**Реализация работы приложения**

Форма входа.

Пользователь вводит данные в форму, если они удовлетворяют всем условиям, то при нажатии кнопки «Регистрация» информация записывается в БД. Для подключения базы данных необходимо создать строку подключения и открыть подключение (SqlConnection).

Чтобы исключить возможность регистрирования пользователей с одинаковым логином, в процессе ввода желаемого имени пользователя, проверяем существует ли уже такой логин в БД. Создаем команду (SqlCommand), которая выведет из БД число пользователей с запрошенным логином

CommandText = "select count(\*) from Users where Login = @login"

В тексте команды используется подготовленное выражение, которое обеспечивает безопасность и быстроту работы.

sCommand.Parameters.AddWithValue("@login", log);

С помощью метода ExecuteScalar() экземпляра класса SqlCommand проверяем уникальность имени пользователя в БД. Если имя уникально, можно подготавливать данные для записи в БД (в противном случае будет выдано соответствующее уведомление).

Для этого необходимо захешировать пароль. Генерируем соль (криптостойкую последовательность случайных значений) и хеш-код для последовательности «соль»+«пароль». Для этого необходимо подключить библиотеку System.Security.Cryptography; Так же в БД по умолчанию занесется статус пользователя «Оператор».

При нажатии на кнопку «Вход» проверяем ввел ли пользователь какие-либо данные в строки логина и пароля. Затем проверяем, существует ли такой пользователь в БД и запрашиваем его статус. В зависимости от статуса меняем заголовок главной формы и видимость кнопки меню «Таблица пользователей». Выводим главную форму на экран.

При нажатии кнопки «Войти как гость» необходимо ограничить возможность пользователя на редактирование данных на главной форме. Для этого при инициализации элементов главной формы устанавливаем следующие свойства:

dgvDis.ReadOnly = true; //запрещает редактирование текста записи

dgvDis.AllowUserToAddRows = false; // запрещает добавление записи

dgvDis.AllowUserToDeleteRows = false; //запрещает удаление записи

Главная форма

На главной форме находится 4 таблицы DataGridView. Для заполнения каждой необходимо создать столбцы. Для этого будем использовать метод Add свойства Column элемента управления. Для столбцов, отвечающих за дату будем использовать следующую конструкцию:

dgvDis.Columns.Add(new CalendarColumn

{

Name = "дата",

HeaderText = @"Дата вакцинации"});

Она позволит вставить DateTimePicker в каждую ячейку соответсвующего столбца.

Для некоторых столбцов логично использовать combobox. Для этого создадим элемент класса DataGridViewComboBoxColumn и добавим его к уже существующим столбцам с помощью метода AddRange.

Теперь, когда все столбцы созданы, заполняем их информацией из БД. Для этого по уже знакомой схеме создаем строку подключения. Создаем и открываем подключение.

Создаем команду (SqlCommand), которая выведет все необходимые данные. И в цикле, пока есть информация, заполняем таблицу построчно. Повторяем эту последовательность действий для каждой таблицы.

Теперь необходимо описать редактирование данных (для пользователя вошедшего как гость эти действия будут недоступны).

Для удаления необходимо описать событие UserDeletingRow элемента управления DataGridView. В теле события создаем команду (SqlCommand), которая удалит выбранную строку из БД и выполняем эту команду с помощью метода ExecuteNonQuery().

Для добавления и изменения используется событие RowValidating. Проверяем существует ли у данной строки id, если он существует, значит происходит удаление, в противном случае добавление. Дальше алгоритм одинаковый: создаем команду (SqlCommand), которая выполнит необходимые действия в БД (INSERT или UPDATE) и выполняем ее.

Если пользователь зашел как администратор для него доступна кнопка меню «Таблица пользователей», при нажатии на которую открывается форма данных о зарегистрированных пользователях.

Форма пользователей

На форме находится элемент ListView, с помощью которого мы сможем представить данные в табличной форме. Необходимо выделить столбцы, для этого будем использовать метод Add свойства Columns экземпляра класса ListView. Создадим 4 столбца: Логин, Пароль, Статус, Дата регистрации (Соль не будем выводить на экран).

Для заполнения таблицы данными необходимо подключение к базе данных. Для подключения базы создадим строку подключения и откроем подключение (SqlConnection). Создадим команду (SqlCommand), которая выведет из БД всю информацию из нужных полей и выполним ее (SqlCommand.ExecuteReader()).

Для того, чтобы заполнить таблицу полученными данными, в цикле создаем новый экземпляр класса ListViewItem и заносим в него данные по столбцам.

lvi = new ListViewItem(new[]

{

(string)reader["Login"],

(string)reader["Password"],

((DateTime)reader["Data"]).ToLongDateString()});

Полю Tag элементов ListViewItem будет присвоено значение даты регистрации. Это значение понадобится нам позже.

lvi.Tag = (DateTime)reader["Data"];

Добавляем новую строку в таблицу используя метод Add свойства Items экземпляра класса ListView.

При запуске полученной программы, мы увидим таблицу, заполненную информацией из БД.

Перейдем к добавлению, изменению и удалению. Создадим «меню». Для этого на форму помещаем элемент управления menuStrip и заносим туда соответствующие названия операций. Создадим еще одну форму, на которую поместим 2 TextBox (для логина и пароля), DateTimePicker (для установки даты регистрации) и кнопку Button (при нажатии которой данные заносятся в БД).

Добавление

При нажатии пользователем на кнопку меню «Добавить» вызывается событие Click элемента добавитьToolStripMenuItem главной формы.

Внутри события создадим экземпляр (formInsert) вспомогательной формы (FormInsert()). Сразу присвоим соответствующий заголовок форме и название кнопке.

formInsert.Text = "Добавить";

formInsert.OK = "Изменить!!!";

Ограничим пользователя в возможности переключаться между формами.

if (formInsert.ShowDialog() == DialogResult.OK)

Процесс добавления состоит из следующих этапов:

* Считываем логин, проверяем на наличие незначащих пробелов и уникальность (в случае ошибок выводим сообщение). Для проверки на уникальность проверяем, существует ли в БД пользователь с таким логином. Делаем это с помощью команды (SqlCommand). Так же стоит проверить поля логина и пароля на пустоту.
* Генерируем соль (криптостойкую последовательность случайных значений) и хеш-код для последовательности «соль»+«пароль». Для этого необходимо подключить библиотеку System.Security.Cryptography;
* Открываем подключение, создаем команду (SqlCommand) добавления данных в БД, выполняем ее. В тексте запроса используются параметры (SqlCommand.Parameters) с целью защиты от SQL-инъекций.

Для вывода только что добавленного элемента на главную форму, снова создаем экземпляр класса ListViewItem, заносим в него данные по столбцам, добавляем его в таблицу с помощью метода Items.Add экземпляра класса ListView. Не забываем заполнить Tag.

При запуске кода на данном этапе видно, что если пользователь ввел неверные данные, то после соответствующего сообщения, форма добавления закрывается и ее придется открывать заново. Это неудобно, поэтому в случае ошибок вызываем функцию добавления рекурсивно.

Удаление

При нажатии пользователем на кнопку меню «Удалить» вызывается событие Click элемента удалитьToolStripMenuItem главной формы.

Удаление организовываем в цикле для каждой выбранной записи из всех присутствующих в таблице.

foreach (ListViewItem selectedItem in lv\_list.SelectedItems)

Внутри цикла открываем подключение, создаем команду (SqlCommand) для удаления записи. Поиск для удаления происходит по уникальному полю Логин. Для получения логина выбранного элемента обращаемся к полю Text элемента selectedItem. Запрос снова организуем с помощью параметров. Выполняем команду используя метод SqlCommand.ExecuteNonQuery().

Для удаления записи с пользовательского интерфейса используем метод Remove свойства Items экземпляра класса ListView, в качестве параметра передается выбранная строка таблицы selectedItem.

lv\_list.Items.Remove(selectedItem);

Изменение

При нажатии пользователем на кнопку меню «Изменение» вызывается событие Click элемента изменитьToolStripMenuItem главной формы. Внутри события создадим экземпляр (formUpdate) вспомогательной формы (FormInsert()). Установим заголовок формы и надпись на кнопке в соответствии с выполняемой задачей.

Далее необходимо занести данные для изменения на форму.

Для того, чтобы получить данные о логине и пароле из таблицы используем свойство SubItems[0].Text экземпляра класса ListViewItem. SubItems[ ] возвращает все подэлементы данного элемента, в скобках указываем номер столбца. Присваиваем полученные значения текстовым полям соответствующих элементов управления. Элементу управления соответствующему дате присвоим значение из Tag-а, явно приведенное к типу DateTime.

formUpdate.Date = (DateTime)selectedItem.Tag;

При нажатии кнопки «Добавить» проверяем на пустоту поля логина и пароля, проверяем логин на уникальность. Важно проверить изменил ли пользователь пароль. Ведь если пароль не был изменен его не нужно заново хешировать. Теперь данные можно вставлять в БД. Снова открываем подключение, создаем команду на вставку используя параметры, выполняем ее используя метод SqlCommand.ExecuteNonQuery().

Осталось только изменить данные в ListView. Для этого опять используем свойство SubItems[0].Text экземпляра класса ListViewItem

selectedItem.SubItems[0].Text = log;

***Тестирование***

Категории тестов:

1. Вход с данными учетной записи со статусом оператор.
2. Вход с данными учетной записи со статусом администратор.
3. Вход с данными несуществующей учетной записи.
4. Вход «как гость».
5. Попытка регистрации с неуникальным логином.
6. Регистрация с уникальным логином и последующий вход.
7. Попытка редактировать данные пользователем «гость».
8. Добавление, удаление и изменение данных пользователями оператор и администратор.
9. Изменение логина и пароля существующей записи администратором.
10. Изменение логина пользователя на другой неуникальный логин администратором.
11. Удаление записи о пользователе администратором.

Для предложенных категорий тестов было выбрано следующее минимальное наполнение БД:

Таблица «Врачи»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id врача | ФИО врача | Телефон | Специализация | Возрастные ограничения |
| 1 | Кабирова Юлия Алексеевна | 89991234567 | Диетолог | детский |

Таблица «Пациенты»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id пациента | ФИО пациента | пол | Дата рождения | телефон |
| 1 | Петров | Мужской | 2017-04-13 | 12345789100 |

Таблица «Запись»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id записи | Id врача | Id пациента | Номер кабинета | Дата | время | примечание |
| 1 | 1 | 1 | 15 | 2017-04-15 | 17:14 | NULL |

Таблицы «Прививки»

|  |  |  |  |
| --- | --- | --- | --- |
| id прививки-пациент | id пациента | id вакцины | Дата |
| 1 | 1 | 1 | 2017-04-14 |

Таблица «Вакцины»

|  |  |
| --- | --- |
| id вакцины | Название |
| 1 | От гриппа |

Таблица «Users»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Login | Password | Salt | Статус | Date |
| 1 | user | 28c2be6e2e164a056ac5e49f934e32250601ed6792c3a2bcf287b620fc48d973 | G90ILvctz+6C+pnfMkIjbJcOER108uxP87BXK8+iTwg= | Оператор | 2017-04-08 |
| 2 | admin | bde19236fa292580b869bc2cd5c3a5d59522df755709db8e1975fead16d41060 | X2uV9298aj1116ACz0ADlc8KvEcMm5Bzf8ic03R//xw= | Администратор | 2017-04-06 |

В таблице представлены хешированные пароли. Исходный пароль для пользователя admin – admin; для пользователя user – user.

Все категории тестов были успешно пройдены.