НИТУ «МИСиС»

Институт ИТАСУ

Кафедра инженерной кибернетики

Направление подготовки: 01.03.04 «прикладная математика»

Квалификация (степень): бакалавр

КУРСОВАЯ РАБОТА

учебная дисциплина

«Специальные главы по базам данных»

VII семестр 2019 – 2020 у.г.

Студент: Малынковский О.В.

Группа: БПМ-16-2

Преподаватель: доц., к.т.н. А.С. Кожаринов

Оценка:

Дата:

Москва 2019

Оглавление

Список используемых сокращений

БД – база данных;

СУБД- система управления базами данных;

ПО – программное обеспечение;

Постановка задачи

Разрабатываемое клиент-серверное приложение относится к предметной области «детский сад»

Любое учреждение дошкольного образования дошкольного образования представляет собой некоторую систему, включающую сотрудников, связи между ними, клиентов (в данном случае родителей и детей), и некоторые элементы, связанные с образовательной деятельностью (например, расписание подготовительных занятий).

Разработка и создание базы данных позволит вести учет всех компонентов системы, обеспечить ее функционирование. Клиент-серверное приложение должно упростить работу с БД как для администрации детского сада, так и дать доступ к необходимой информации родителям в режиме онлайн.

Для администрации важно иметь возможностью управлять кадровым составом своего учреждения, поэтому необходимо включить возможностью изменять данные о сотрудниках в приложении. Также обязательно наличие инструментов создания и редактирования расписания для каждой группы детей. Отдельным моментов является возможность вести учет оплаты родителями за оказываемые в детском саду образовательные услуги.

Для родителей также важно обеспечить доступ к информации о себе и своих детей, особенно это касается наличия или отсутствии долга по оплате. Также должен предоставляться доступ к расписанию занятий своих детей.

.

Использованные средства разработки и системные требования

Наиболее удобным и подходящим средством для работы с БД является использование десктопного (оконного) приложения. Исходя из такой задачи средством разработки программного продукта станет Microsoft Visual Studio. СУБД - MySQL. В качестве языка программирования будет использоваться C#. Для создания графического интерфейса приложения задействуем средства WPF.

Приложение должно работать в Windows 10.

Модель данных

Структура модели

При проведении анализа предметной области были выделины следующие сущности, которые образуют концептуальную модель данных подсистемы: родители, дети, учителя, воспитатели, нянечки, группы, бухгалтеры, администраторы, квитанции об оплате, расписание. На рис. 2. представлена даталогическая модель.

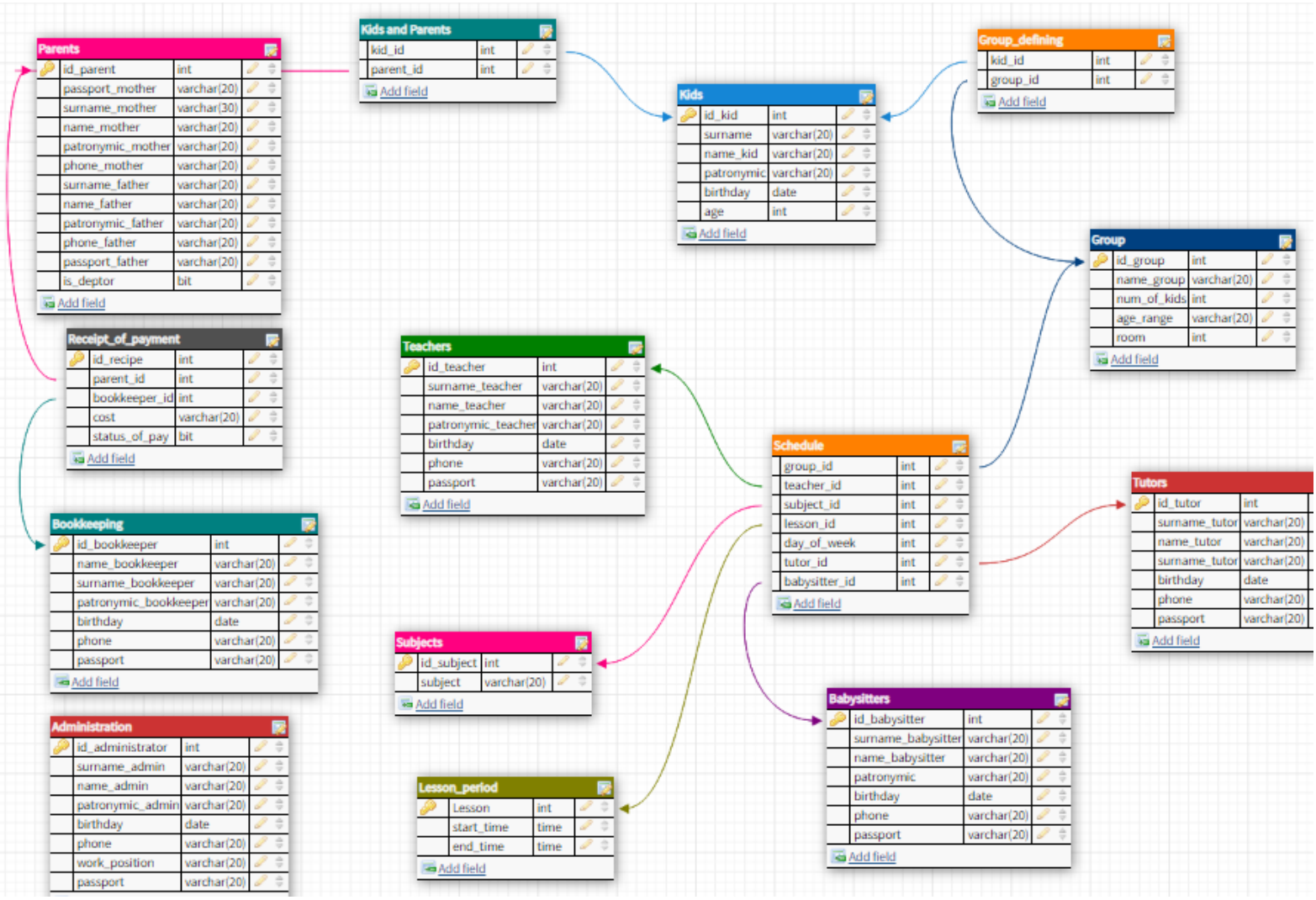


Рисунок 1.

Таблицы базы данных

Приведем общий список таблиц, созданных учащимся в БД.

№	Название таблицы	Назначение
1	Parents	Данные о родителях
2	Kids	Данные о детях

3	Kids_and_parents	Связь родителей и детей
4	Schedule	Данные о расписании
5	Subjects	Данные о изучаемых предметах
6	Teachers	Данные о педагогах
7	Tutors	Данные о воспитателях
8	Babysitters	Данные о нянях
9	Groups	Данные о группах
10	Group_defining	Связь детей и групп
11	Lesson_period	Данные о времени звонков
12	Bookkeeping	Данные о бухгалтерах
13	Recipe_of_payment	Данные об оплате
14	Administration	Данные об администрации

После общего списка учащийся приводит краткое описание структуры каждой таблицы в следующей форме:

Parents				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_parent	Int	PK	Идентификатор родителя
2	Surname_mother	varchar		Фамилия мамы
3	Name_mother	varchar		Имя мамы
4	Patronymic_mother	varchar		Отчество мамы
5	Passport_mother	varchar		Паспорт мамы
6	Phone_mother	varchar		Телефон мамы
7	Surname_father	varchar		Фамилия папы

8	Name_father	varchar		Имя папы
9	Patronymic_father	varchar		Отчество папы
10	Passport_father	varchar		Паспорт папы
11	Phone_father	varchar		Телефон папы
12	Is_deptor	bit		Должны по оплате или нет

Kids				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_kid	Int	PK	Идентификатор ребенка
2	Surname	Varchar		Фамилия ребенка
3	Name_kid	Varchar		Имя ребенка
4	Patronymic	Varchar		Отчество ребенка
5	Birthday	Date		Дата рождения
6	Age	Int		Возраст

Kids_and_parents				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Kid_id	Int	FK	Идентификатор ребенка
2	Parent_id	Int	FK	Идентификатор родителя

Schedule				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_Schedule	Int	PK	Идентификатор строки расписания
2	Group_id	Int	FK	Идентификатор группы
3	Day_of_week	Int		День недели
4	Lesson_id	Int	FK	Идентификатор урок (его времени проведения)
5	Subject_id	Int	FK	Идентификатор предмета
6	Teacher_id	Int	FK	Идентификатор учителя
7	Tutor_id	Int	FK	Идентификатор воспитателя
8	Babysitter_id	Int	FK	Идентификатор няни

Subjects				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_subject	Int	PK	Идентификатор предмета
2	Name_of_subject	Varchar		Название предмета

Teachers				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_teacher	int	PK	Идентификатор учителя

2	Surname_teacher	Varchar		Фамилия учителя
3	Name_teacher	Varchar		Имя
4	Patronymic_teacher	Varchar		Отчество
5	Passport	Varchar		Паспорт
6	Phone	Varchar		Номер телефона
7	Birthday	Date		Дата рождения
9	Subject_id	Int	FK	Идентификатор преподаваемого предмета

Tutors				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_tutor	int	PK	Идентификатор воспитателя
2	Surname_ tutor	Varchar		Фамилия
3	Name_ tutor	Varchar		Имя
4	Patronymic_tutor	Varchar		Отчество
5	Passport	Varchar		Паспорт
6	Phone	Varchar		Номер телефона
7	Birthday	Date		Дата рождения

Babysitters				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение

1	Id_babysitter	int	PK	Идентификатор няни
2	Surname_ babysitter	Varchar		Фамилия
3	Name_ babysitter	Varchar		Имя
4	Patronymic_ babysitter	Varchar		Отчество
5	Passport	Varchar		Паспорт
6	Phone	Varchar		Номер телефона
7	Birthday	Date		Дата рождения

Groups				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_group	Int	PK	Идентификатор группы
2	Name_group	Varchar		Название группы
3	Num_of_kid	Int		Количество детей
4	Age_range	Varchar		Возрастная категория
5	Room	Int		Кабинет

Group_defining				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Kid_id	Int	FK	Идентификатор ребенка
2	Group_id	Int	FK	Идентификатор группы

Lesson_period				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_lesson	Int	PK	Идентификатор урока
2	Start_time	Time		Время начала урока
3	End_time	Time		Время окончания

Bookkeeping				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_bookkeeper	int	PK	Идентификатор бухгалтера
2	Surname_ bookkeeper	Varchar		Фамилия
3	Name_ bookkeeper	Varchar		Имя
4	Patronymic_ bookkeeper	Varchar		Отчество
5	Passport	Varchar		Паспорт
6	Phone	Varchar		Номер телефона
7	Birthday	Date		Дата рождения

Recipe_of_payment				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_recipe	Int	PK	Идентификатор квинтанции об оплате

2	Parent_id	Int	FK	Идентификатор родителя
3	Bookkeeper_id	Int	FK	Идентификатор бухгалтера
4	Cost	Varchar		Сумма к оплате
5	Status_of_pay	Bit		Статус платежа

Administration				
№	Название поля	Тип данных	Ключ (PK/FK)	Назначение
1	Id_administrator	int	PK	Идентификатор бухгалтера
2	Surname_ administrator	Varchar		Фамилия
3	Name_ administrator	Varchar		Имя
4	Patronymic_ administrator	Varchar		Отчество
5	Passport	Varchar		Паспорт
6	Phone	Varchar		Номер телефона
7	Birthday	Date		Дата рождения
8	Work_position	Varchar		Должность

Также созданы триггеры для таблиц учителей, воспитателей и нянь, которые заменяет данные в строчках расписания на «Вакансия», если будут удалены сотрудники, которые уже задействованы в учебном расписании. Для таблицы с квитанциями об оплате создан триггер, автоматически меняющий статус платежа в зависимости от указанной суммы к оплате.

Описание клиентского приложения

Общие сведения

Клиент-серверное приложение должно упростить работу с БД как для администрации детского сада, так и дать доступ к необходимой информации родителям в режиме онлайн.

Для администрации важно иметь возможностью управлять кадровым составом своего учреждения, поэтому необходимо включить возможностью изменять данные о сотрудниках в приложении. Также обязательно наличие инструментов создания и редактирования расписания для каждой группы детей. Отдельным моментов является возможность вести учет оплаты родителями за оказываемые в детском саду образовательные услуги.

Для родителей также важно обеспечить доступ к информации о себе и своих детей, особенно это касается наличия или отсутствии долга по оплате. Также должен предоставляться доступ к расписанию занятий своих детей.

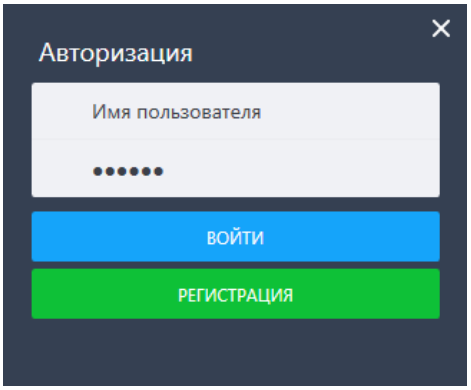
При входе нужно пройти аутентификацию через ввод логина и пароля. Для родителей есть возможность самостоятельно зарегистрироваться в системе. Для этого ин нужно ввести личные данные и придумать надежный пароль. Администрация не может самостоятельно регистрироваться, их учетные данные уже есть в системе.

Список классов, разработанных для работы приложения

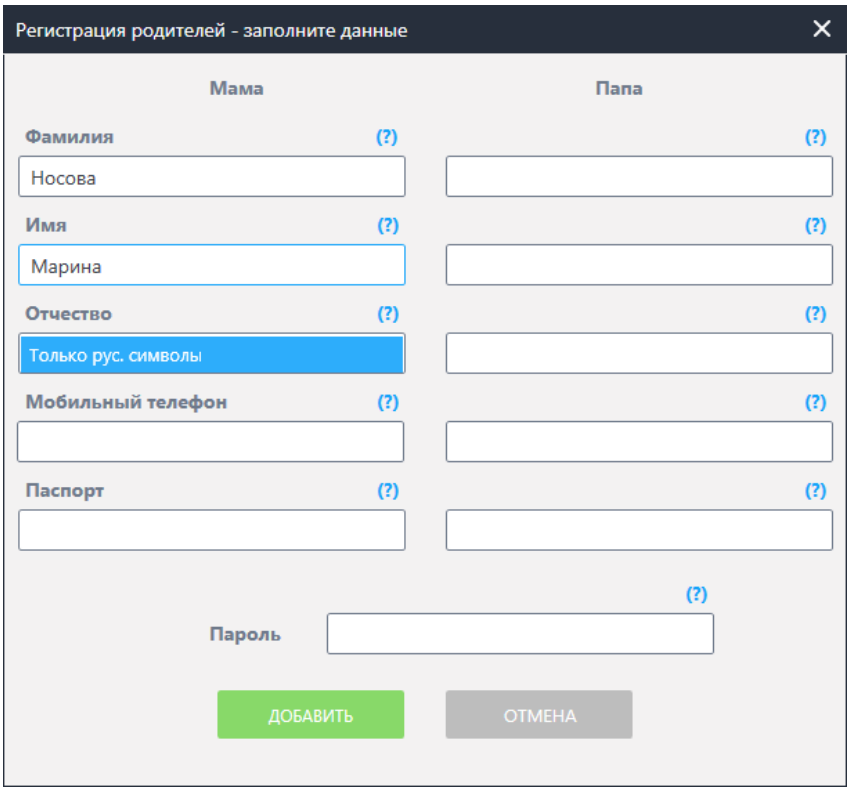
public partial class MainWindow : Window – главное окно приложения, все основные функции
public partial class LoginWindow : Window – окно авторизации
public partial class NewClientWindow : Window – окно регистрации
public class DB - класс, где реализована работа с подключенной БД
class Checks – проверка вводимого текста на соответствие формату
class Animation – анимации переходов между окнами

Описание экранных форм

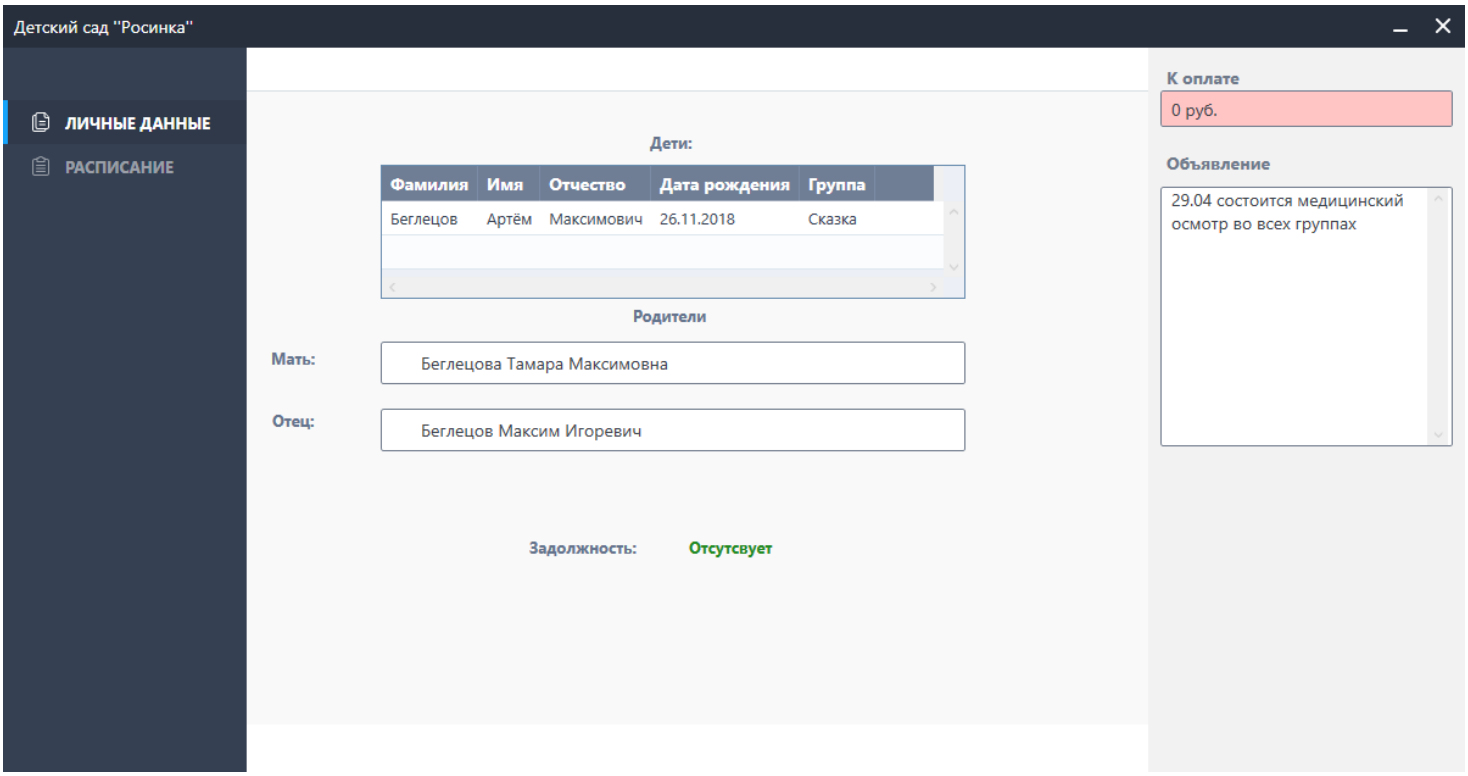
При запуске приложения открывается форма для ввода учетных данных. Если пользователь впервые открывает приложение, то ему необходимо нажать на кнопку регистрации, чтобы получить логин и пароль для входа.



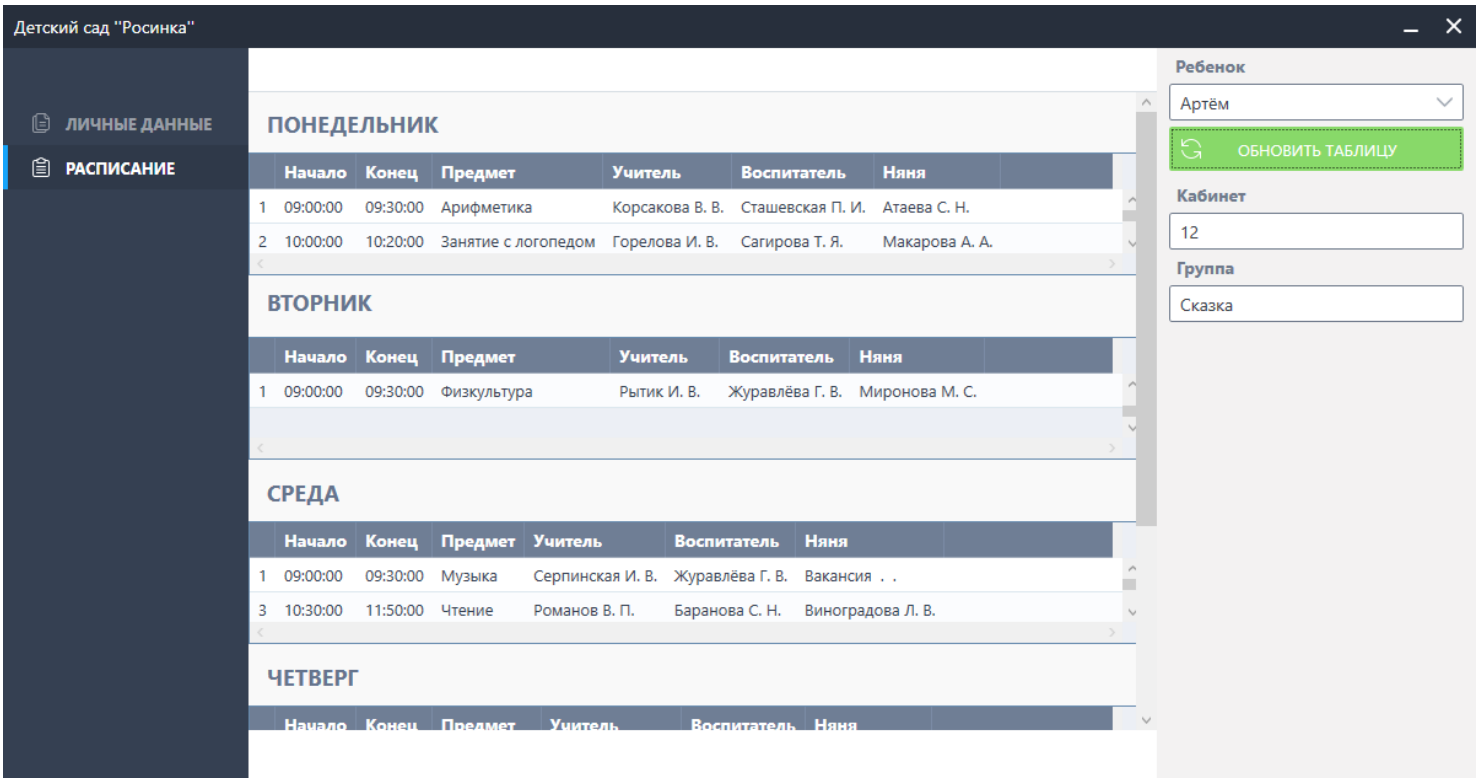
Регистрируются только родители. Сотрудники администрации уже имеют учетные данные в системе. Родителям необходимо ввести информацию о себе и придумать надежный пароль. Введенные данные сверяются с данными в БД, и в случае нахождения совпадения пользователю выдается логин. После этого уже можно войти в систему.



Вид приложения отличается функционал для родителей и администрации. У родителей слева есть две кнопки: личные данные и расписание. В первой вкладке выводятся основные сведениях о родителях и их детях. Также указана информация о том есть ли долг по оплате. Справа отображаются сумма к оплате и поле с важными объявлениями.

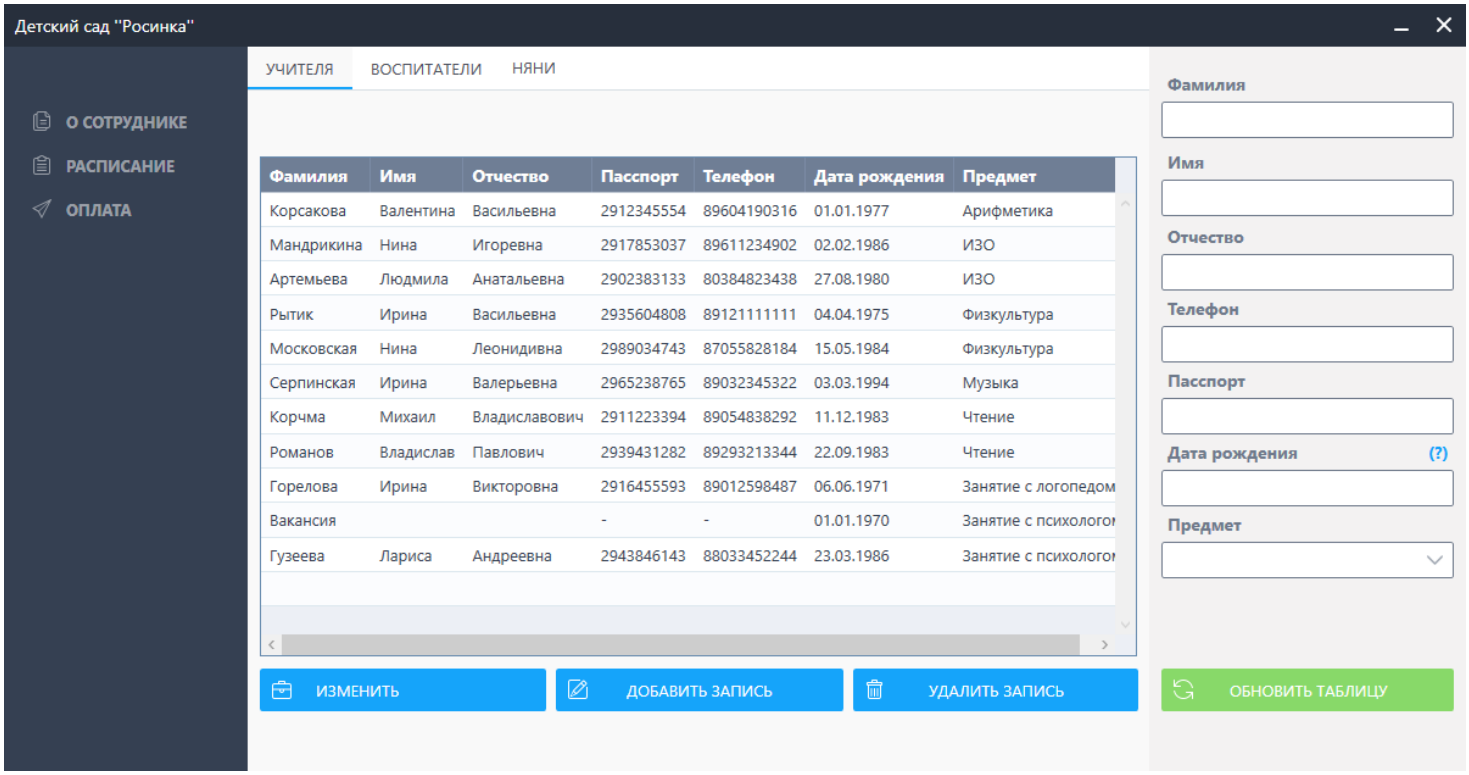


Во вкладке расписания можно увидеть по дням список всех занятии проводимых в группе. Справа можно выбрать ребенка, если их более одного. Эта функция нужна, если дети в разных группах, и у них, соответственно, разные расписания занятий. Также справа указаны название группы и номер кабинета.

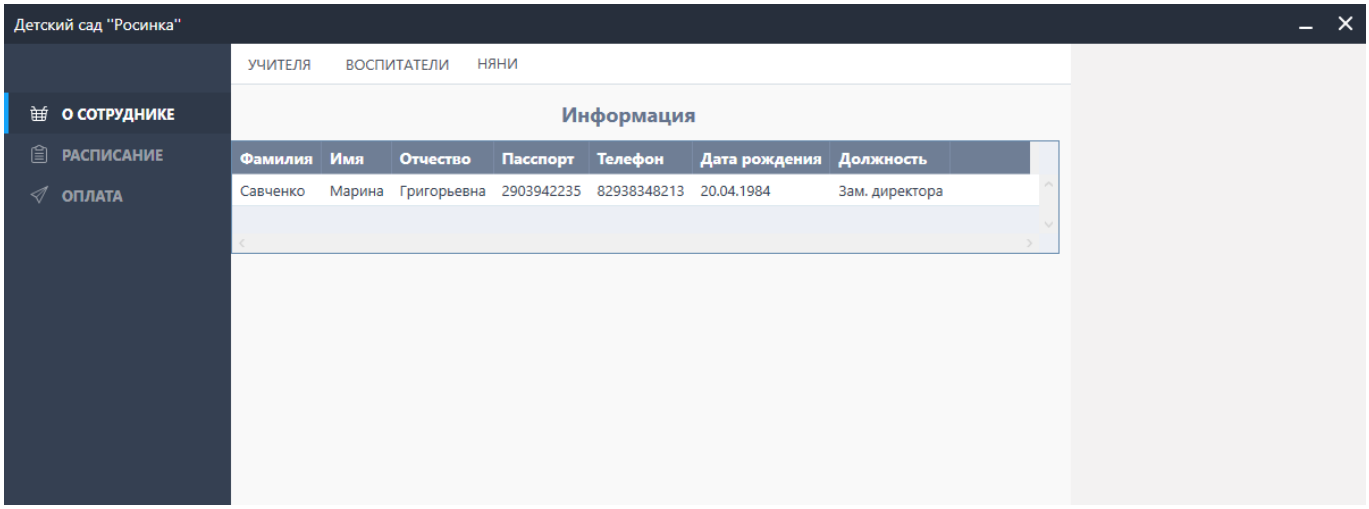


Если в систему входит сотрудник администрации, то у него есть вкладки: о сотруднике, расписание, оплата слева, а справа учителя, воспитатели, няни.

Открывая вкладку «учителя», отображается весь список преподавателей. Выбирая строчку в таблице, можно удалить ее или прямо там изменить данные (например, у человека может поменяться номер телефона) с помощью кнопок под таблицей. Если нужно добавить нового сотрудника, то справа вводятся все данные а потом нажимается кнопка добавить запись.

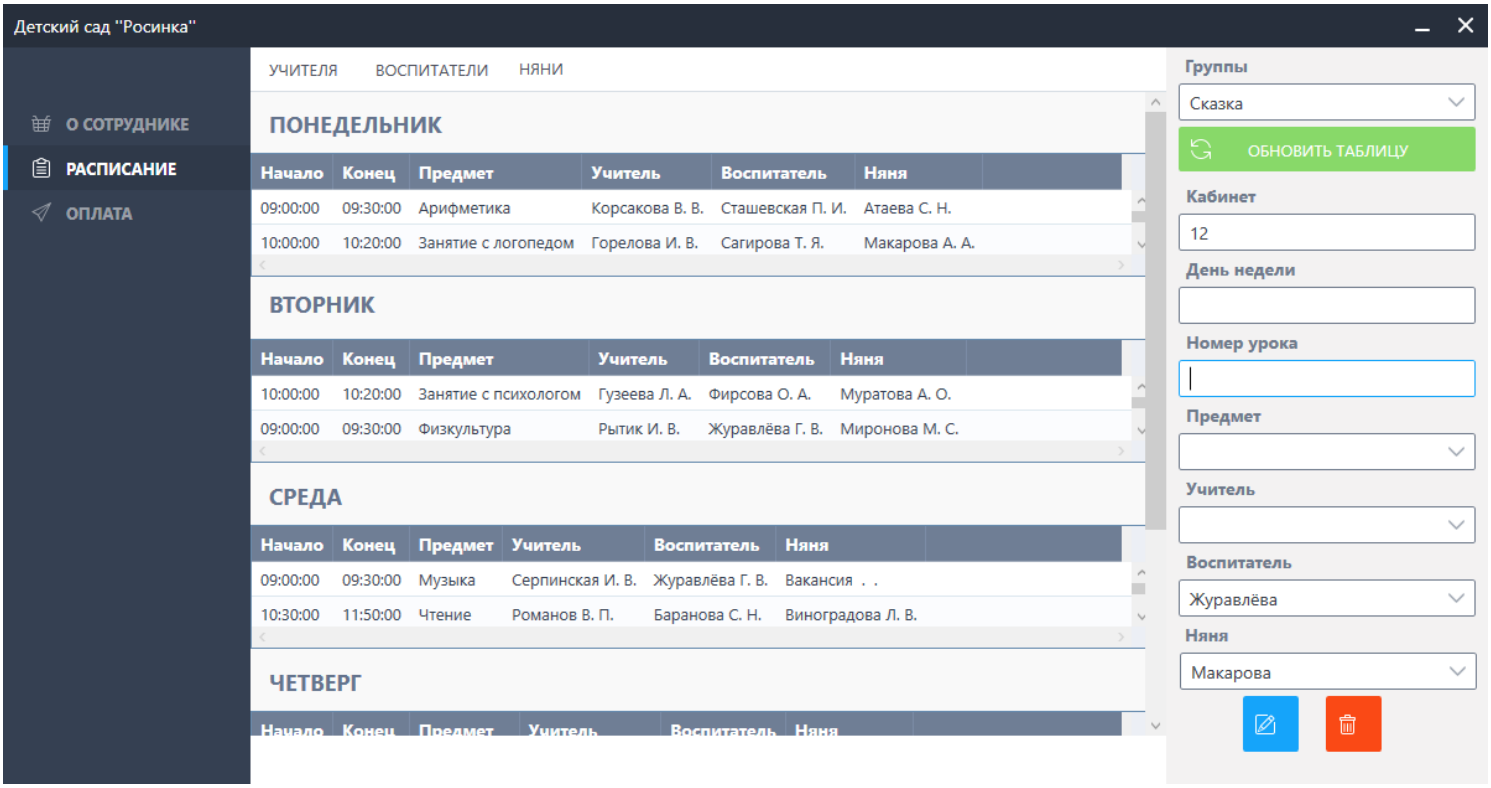


Во вкладке «о сотруднике» выводятся данные о пользователе системы.

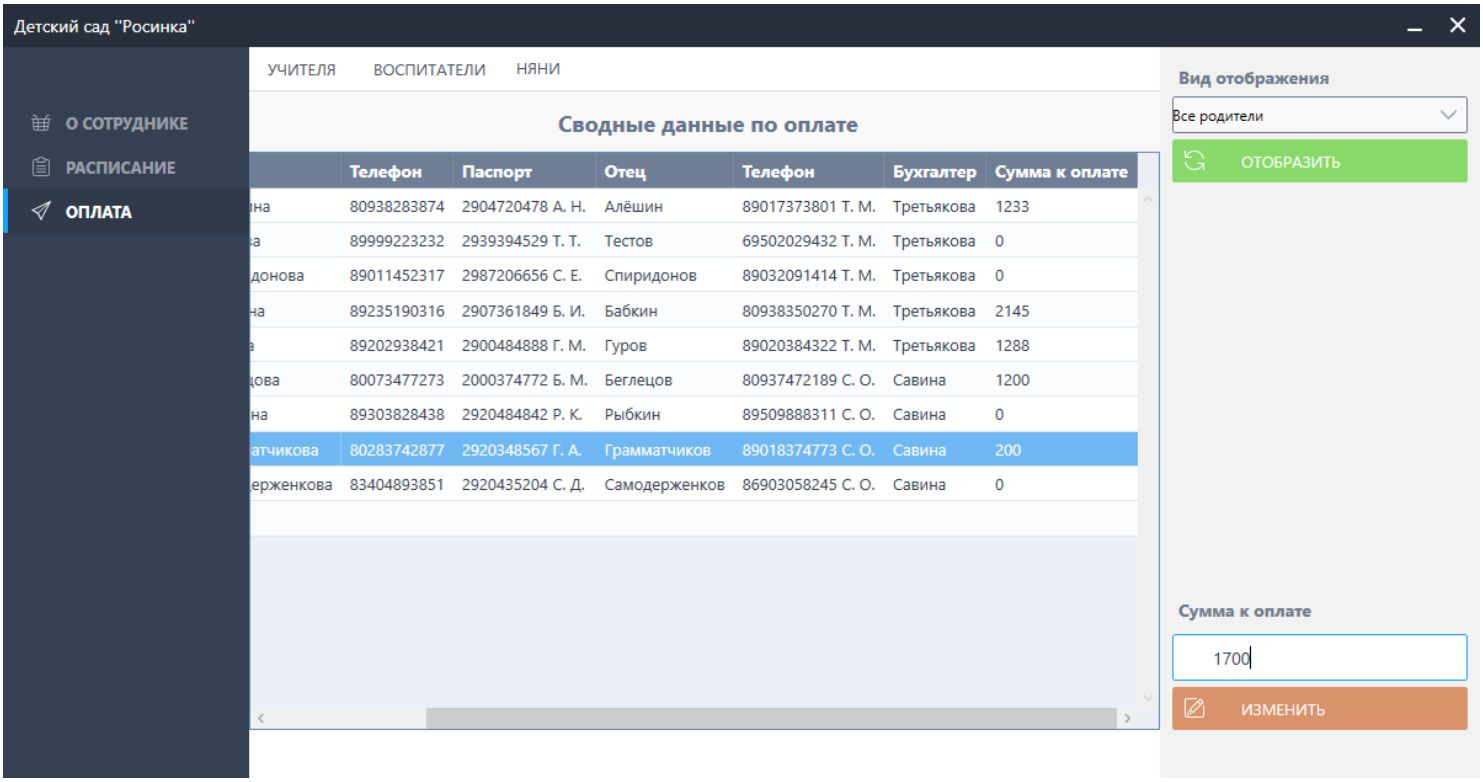


Во вкладке «расписание» можно редактировать занятия для всех групп. Справа можно выбрать ту группу, которую нужно отобразить. Для удаления занятие нужно выбрать строку в

таблице, а затем нажать справа внизу оранжевую кнопку. Справа можно ввести данные и с помощью синей кнопки ниже добавить занятие в расписание этой группы.



Во вкладке «оплата» выводятся все данные о родителях и сумму к оплате. Можно изменить значение поля суммы к оплате. Также для удобства можно включить вывод всех или только должников.



Организация ролевого доступа к данным.

Реализованы две роли, за исключение администратора БД, имеющего полный доступ ко всей БД.

Первая роль – родитель. Он имеет возможность только просматривать содержимое таблиц без права какой-либо модификации строк и самих таблиц. Чтобы пользователю не выводилась строки с чужой информацией (таблица с родителями, где есть номера паспортов и телефонов) в коде самого приложения организованы запросы только к допустимым данным.

Вторая роль – сотрудник администрации. У него есть права на модификацию строк во всех таблицах (но в приложении это реализовано не для всех таблиц),однако удалять сами таблицы или добавить новые он не в праве

Тематические запросы к данным

Получение отчета об оплате для всех родителей:

select id_parent, passport_mother, surname_mother, name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father, patronymic_father, phone_father, surname_bookkeeper, name_bookkeeper,patronymic_bookkeeper, cost from parents join recipe_of_payment rop on parents.id_parent = rop.parent_id join bookkeeping b on rop.bookkeeper_id = b.id_bookkeeper

Аналогичное действие, но только для родителей с долгом по оплате:

select id_parent, passport_mother, surname_mother, name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father, patronymic_father, phone_father, surname_bookkeeper, name_bookkeeper,patronymic_bookkeeper, cost from parents join recipe_of_payment rop on parents.id_parent = rop.parent_id join bookkeeping b on rop.bookkeeper_id = b.id_bookkeeper where status_of_pay = false

Выводы

В рамках данной курсовой работы выполнен системный анализ предметной области, разработана структура системы, модели данных. Реализовано пользовательское приложение с разделением ролей в системе и удобным графическим интерфейсом.

Приложения

SQL-скрипт создания БД

```
CREATE TABLE android_test.Tutors(
	id_tutor int AUTO_INCREMENT PRIMARY KEY,
	surname_tutor varchar(30),
	name_tutor varchar(20)
	patronymic_tutor varchar(20),
	passport varchar(20),
	phone varchar(20),
	birthday date );
CREATE TABLE android_test.Subjects(
	id_subject int AUTO_INCREMENT PRIMARY KEY,
	name_of_subject varchar(20));
CREATE TABLE android_test.Lesson_period(
	id_lesson int AUTO_INCREMENT PRIMARY KEY,
	start_time time,
	end_time time);
CREATE TABLE android_test.Parents(
	id_parent int AUTO_INCREMENT PRIMARY KEY,
	passport_mother varchar(20),
	surname_mother varchar(30),
	name_mother varchar(20),
	patronymic_mother varchar(20),
	phone_mother varchar(20),
	passport_father varchar(20),
	surname_father varchar(30),
	name_father varchar(20),
	patronymic_father varchar(20),
	phone_father varchar(20),
	is_deptor bit);
CREATE TABLE android_test.Bookkeeping(
	id_bookkeeper int AUTO_INCREMENT PRIMARY KEY,
	surname_bookkeeper varchar(30),
	name_bookkeeper varchar(20),
	patronymic_bookkeeper varchar(20),
	passport varchar(20),
	phone varchar(20),
	birthday date,
	work_position nvarchar(20));
CREATE TABLE android_test.Babysitters(
	id_babysitter int AUTO_INCREMENT PRIMARY KEY,
	surname_babysitter varchar(30),
	name_babysitter varchar(20),
	patronymic_babysitter varchar(20),
	passport varchar(20),
	phone varchar(20),
	birthday date );
CREATE TABLE android_test.Administration(
	id_administrator int AUTO_INCREMENT PRIMARY KEY,
	surname_administrator varchar(30),
```

```
        name_administrator varchar(20),
        patronymic_administrator varchar(20),
        passport varchar(20),
        phone nvarchar(20),
        birthday date,
        work_position nvarchar(20) );
CREATE TABLE android_test.Kids(
        id_kid int AUTO_INCREMENT PRIMARY KEY,
        surname varchar(30),
        name_kid varchar(20),
        patronymic varchar(20),
        birthday date,
        age int,
        is_sick bit);
CREATE TABLE android_test.Groups_(
        id_group int AUTO_INCREMENT PRIMARY KEY,
        name_group varchar(20),
        num_of_kid int,
        age_range varchar(3) );

CREATE TABLE android_test.Group_defining(
        kid_id int,
        group_id int,
   FOREIGN KEY (kid_id) REFERENCES android_test.Kids (id_kid) ON DELETE CASCADE,
   FOREIGN KEY (group_id) REFERENCES android_test.Groups_ (id_group) ON DELETE
CASCADE);
CREATE TABLE android_test.Recipe_of_payment(
        id_recipe int AUTO_INCREMENT PRIMARY KEY,
        parent_id int,
        bookkeeper_id int,
        cost varchar(50),
        status_of_pay bit,
   FOREIGN KEY (parent_id) REFERENCES android_test.Parents (id_parent) ON DELETE
CASCADE,
   FOREIGN KEY (bookkeeper_id) REFERENCES android_test.Bookkeeping (id_bookkeeper)
ON DELETE CASCADE);
CREATE TABLE android_test.Kids_and_Parents(
        kid_id int NOT NULL,
        parent_id int NOT NULL,
   FOREIGN KEY (parent_id) REFERENCES android_test.Parents (id_parent) ON DELETE
CASCADE,
   FOREIGN KEY (kid_id) REFERENCES android_test.Kids (id_kid) ON DELETE
CASCADE);
CREATE TABLE android_test.Teachers(
        id_teacher int AUTO_INCREMENT PRIMARY KEY,
        surname_teacher varchar(30),
        name_teacher varchar(20),
        patronymic_teacher varchar(20),
        passport varchar(20),
        phone varchar(20) ,
        birthday date,
        id_subject int,
```

```
   FOREIGN KEY (id_subject) REFERENCES android_test.Subjects (id_subject) ON DELETE
CASCADE);
CREATE TABLE  android_test.Schedule(
        group_id int,
        day_of_week int,
        lesson_id int,
        subject_id int,
        teacher_id int,
        tutor_id int,
        babysitter_id int,
        id_Schedule int AUTO_INCREMENT PRIMARY KEY,
   FOREIGN KEY (group_id) REFERENCES android_test.Groups_ (id_group) ON DELETE
NO ACTION,
   FOREIGN KEY (lesson_id) REFERENCES android_test.Lesson_period (id_lesson) ON
DELETE NO ACTION,
   FOREIGN KEY (subject_id) REFERENCES android_test.Subjects (id_subject) ON DELETE
NO ACTION,
   FOREIGN KEY (event_id) REFERENCES android_test.Events_for_kids (id_event) ON
DELETE NO ACTION,
   FOREIGN KEY (teacher_id) REFERENCES android_test.Teachers (id_teacher) ON DELETE
NO ACTION,
   FOREIGN KEY (tutor_id) REFERENCES android_test.Tutors (id_tutor) ON DELETE NO
ACTION,
   FOREIGN KEY (babysitter_id) REFERENCES android_test.Babysitters (id_babysitter) ON
DELETE NO ACTION
);
```

Триггеры:

```
CREATE TRIGGER DEL_1 AFTER DELETE ON babysitters FOR EACH ROW
BEGIN
   update schedule set babysitter_id = 7 where schedule.babysitter_id is null;
END;
CREATE TRIGGER DEL_2 AFTER DELETE ON teachers FOR EACH ROW
BEGIN
   update schedule set teacher_id = 7 where schedule.teacher_id is null;
END;
CREATE TRIGGER DEL_3 AFTER DELETE ON tutors FOR EACH ROW
BEGIN
   update schedule set tutor_id = 7 where schedule.tutor_id is null;
END;

CREATE TRIGGER PAY_UPD before update ON recipe_of_payment FOR EACH ROW
BEGIN
   if (NEW.cost != 0) then set NEW.status_of_pay = false; end if;
   if (NEW.cost = 0) then set NEW.status_of_pay = true; end if;
END;
```

Заполнение БД:

```
INSERT INTO android_test.shift_period (id_shift, name_of_shift, start_time, end_time)
VALUES (1, 'Утренняя', '07:30:00', '12:30:00')
```

INSERT INTO android_test.shift_period (id_shift, name_of_shift, start_time, end_time) VALUES (2, 'Дневная', '12:30:00', '17:30:00')
INSERT INTO android_test.subjects (name_of_subject) VALUES ('Арифметика')
INSERT INTO android_test.subjects (name_of_subject) VALUES ('ИЗО')
INSERT INTO android_test.subjects (name_of_subject) VALUES ('Физкультура')
INSERT INTO android_test.subjects (name_of_subject) VALUES ('Музыка')
INSERT INTO android_test.subjects (name_of_subject) VALUES ('Чтение')
INSERT INTO android_test.subjects (name_of_subject) VALUES ('Занятие с логопедом')
INSERT INTO android_test.subjects (name_of_subject) VALUES ('Занятие с психологом')

INSERT INTO android_test.teachers (id_teacher, surname_teacher, name_teacher, patronymic_teacher, passport, phone, birthday, id_subject) VALUES (1, 'Корсакова', 'Валентина', 'Васильевна', 2912345554, '89604190316', '1977.01.01', 1)

INSERT INTO android_test.teachers (id_teacher, surname_teacher, name_teacher, patronymic_teacher, passport, phone, birthday, id_subject) VALUES (2, 'Мандрикина', 'Нина', 'Игоревна', 2917853037, '89611234902', '1986.02.02', 2)

INSERT INTO android_test.teachers (id_teacher, surname_teacher, name_teacher, patronymic_teacher, passport, phone, birthday, id_subject) VALUES (3, 'Серпинская', 'Ирина', 'Валерьевна', 2965238765, '89032345322', '1994.03.03', 4)

INSERT INTO android_test.teachers (id_teacher, surname_teacher, name_teacher, patronymic_teacher, passport, phone, birthday, id_subject) VALUES (4, 'Рытик', 'Ирина', 'Васильевна', 2935604808, '89121111111', '1975.04.04', 3)

INSERT INTO android_test.teachers (id_teacher, surname_teacher, name_teacher, patronymic_teacher, passport, phone, birthday, id_subject) VALUES (5, 'Корчма', 'Михаил', 'Владиславович', 2911345433, '89123333333', '1988.05.05', 5)

INSERT INTO android_test.teachers (id_teacher, surname_teacher, name_teacher, patronymic_teacher, passport, phone, birthday, id_subject) VALUES (6, 'Горелова', 'Ирина', 'Викторовна', 2916455593, '89012598487', '1971.06.06', 6)

INSERT INTO android_test.teachers (id_teacher, surname_teacher, name_teacher, patronymic_teacher, passport, phone, birthday, id_subject) VALUES (7, 'Терехова', 'Наталья', 'Львовна', 2906382939, '89999999999', '1989.07.07', 7)

INSERT INTO android_test.lesson_period (id_lesson, start_time, end_time) VALUES (1, '09:00:00', '09:30:00')

INSERT INTO android_test.lesson_period (id_lesson, start_time, end_time) VALUES (2, '10:00:00', '10:20:00')

INSERT INTO android_test.lesson_period (id_lesson, start_time, end_time) VALUES (3, '10:30:00', '11:50:00')

INSERT INTO android_test.lesson_period (id_lesson, start_time, end_time) VALUES (4, '15:35:00', '16:00:00')

INSERT INTO android_test.tutors (surname_tutor, name_tutor, patronymic_tutor, passport, phone, birthday) VALUES ('Сидоренко', 'Марина', 'Юрьевна', '2918272673', '89001112233', '1993.07.30')

INSERT INTO android_test.tutors (surname_tutor, name_tutor, patronymic_tutor, passport, phone, birthday) VALUES ('Фролов', 'Денис', 'Максимович', '2911234432', '88005553535', '1987.08.31')

INSERT INTO android_test.tutors (surname_tutor, name_tutor, patronymic_tutor, passport, phone, birthday) VALUES ('Журавлёва', 'Галина', 'Викторовна', '2979797979', '89001112233', '1970.09.11')

INSERT INTO android_test.tutors (surname_tutor, name_tutor, patronymic_tutor, passport, phone, birthday) VALUES ('Сагирова', 'Татьяна', 'Яковлевна', '2912324354', '81235676433', '1982.10.10')

INSERT INTO android_test.tutors (surname_tutor, name_tutor, patronymic_tutor, passport, phone, birthday) VALUES ('Баранова', 'Светлана', 'Николаевна', '2192838485', '87084475754', '1985.05.25')

INSERT INTO android_test.tutors (surname_tutor, name_tutor, patronymic_tutor, passport, phone, birthday) VALUES ('Фирсова', 'Ольга', 'Анатольевна', '2944534531', '80047777221', '1992.03.08')

INSERT INTO android_test.technical_staff (surname_staff, name_staff, patronymic_staff, passport, phone, birthday, work_position, shift_id) VALUES ('Фролова', 'Татьяна', 'Васильевна', 2914080145, '89041245890', '1977.08.31', 'повар', 1)

INSERT INTO android_test.technical_staff (surname_staff, name_staff, patronymic_staff, passport, phone, birthday, work_position, shift_id) VALUES ('Бонз', 'Дмитрий', 'Олегович', 2917432290, '89704245281', '1972.09.11', 'повар', 2)

INSERT INTO android_test.technical_staff (surname_staff, name_staff, patronymic_staff, passport, phone, birthday, work_position, shift_id) VALUES ('Никольская', 'Виктория', 'Владимировна', 2918554381, '80923747487', '1969.01.14', 'помощник повара', 1)

INSERT INTO android_test.technical_staff (surname_staff, name_staff, patronymic_staff, passport, phone, birthday, work_position, shift_id) VALUES ('Смирнова', 'Яна', 'Николаевна', 2913506433, '89079073454', '1965.12.12', 'помощник повара', 2)

INSERT INTO android_test.technical_staff (surname_staff, name_staff, patronymic_staff, passport, phone, birthday, work_position, shift_id) VALUES ('Фирсова', 'Татьяна', 'Владимировна', 2918475483, '89042838458', '1970.12.14', 'уборщица', 1)

INSERT INTO android_test.technical_staff (surname_staff, name_staff, patronymic_staff, passport, phone, birthday, work_position, shift_id) VALUES ('Карев', 'Игорь', 'Андреевич', 2910384383, '89027263647', '1968.07.21', 'уборщик', 2)

INSERT INTO android_test.parents (passport_mother, surname_mother, name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father, patronymic_father, phone_father, is_deptor) VALUES ('2911300200', 'Спиридонова', 'Наталья', 'Алексеевна', '89011452317', '2987206656', 'Спиридонов', 'Егор', 'Алексеевич', '89032091414', false)

INSERT INTO android_test.parents (passport_mother, surname_mother, name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father, patronymic_father, phone_father, is_deptor) VALUES ('2908074662', 'Бабкина', 'Виктория', 'Дмитриевна', '89235190316', '2907361849', 'Бабкин', 'Иван', 'Юрьевич', '80938350270', false)

INSERT INTO android_test.parents (passport_mother, surname_mother, name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father, patronymic_father, phone_father, is_deptor) VALUES ('2847402749', 'Алёшина', 'Татьяна', 'Александровна', '80938283874', '2904720478', 'Алёшин', 'Никита', 'Эдуардович', '89017373801', false)

INSERT INTO android_test.parents (passport_mother, surname_mother, name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father, patronymic_father, phone_father, is_deptor) VALUES ('2886731139', 'Беглецова', 'Тамара', 'Максимовна', '80073477273', '2000374772', 'Беглецов', 'Максим', 'Игоревич', '80937472189', false)

INSERT INTO android_test.doctors (surname_doctor, name_doctor, patronymic, phone, passport, birthday) VALUES ('Смирнова', 'Светлана', 'Андреевна', '89705441020', 2911004365, '1985.02.11')

INSERT INTO android_test.doctors (surname_doctor, name_doctor, patronymic, phone, passport, birthday) VALUES ('Гагарина', 'Тамара', 'Львовна', '89402345644', 2908343244, '1973.08.28')

INSERT INTO android_test.bookkeeping (surname_bookkeeper, name_bookkeeper, patronymic_bookkeeper, passport, phone, birthday, work_position) VALUES ('Третьякова', 'Мария', 'Геннадьевна', '2913676224', '89001115532', '1993.11.25', 'Старший бухгалтер')

INSERT INTO android_test.bookkeeping (surname_bookkeeper, name_bookkeeper, patronymic_bookkeeper, passport, phone, birthday, work_position) VALUES ('Савина', 'Ольга', 'Алексеевна', '2901343433', '80232448348', '1995.08.13', 'Бухгалтер')

INSERT INTO android_test.administration (surname_administrator, name_administrator, patronymic_administrator, passport, phone, birthday, work_position) VALUES ('Дьячук', 'Ирина', 'Борисовна', '2912345443', '89073242244', '1970.01.19', 'Директор')

INSERT INTO android_test.administration (surname_administrator, name_administrator, patronymic_administrator, passport, phone, birthday, work_position) VALUES ('Савченко', 'Марина', 'Григорьевна', '2903942235', '82938348213', '1984.04.20', 'Зам. директора')

INSERT INTO android_test.administration (surname_administrator, name_administrator, patronymic_administrator, passport, phone, birthday, work_position) VALUES ('Кирсанова', 'Яна', 'Олеговна', '2004838234', '89038472838', '1988.05.15', 'Заведующая уч.частью')

INSERT INTO android_test.administration (surname_administrator, name_administrator, patronymic_administrator, passport, phone, birthday, work_position) VALUES ('Маслова', 'Инна', 'Сергеевна', '2903902381', '80923828123', '1980.09.22', 'Заведующая хоз.частью')

INSERT INTO android_test.babysitters (surname_babysitter, name_babysitter, patronymic_babysitter, passport, phone, birthday) VALUES ('Прокопьева', 'Светлана', 'Викторовна', 2910234341, '89014139845', '1996.07.27')

INSERT INTO android_test.babysitters (surname_babysitter, name_babysitter, patronymic_babysitter, passport, phone, birthday) VALUES ('Семашко', 'Ангелина', 'Борисовна', 2910384323, '89323929292', '1992.08.21')

INSERT INTO android_test.babysitters (surname_babysitter, name_babysitter, patronymic_babysitter, passport, phone, birthday) VALUES ('Ермакова', 'Яна', 'Григорьевна', 2910238821, '89038329113', '1990.11.03')

INSERT INTO android_test.babysitters (surname_babysitter, name_babysitter, patronymic_babysitter, passport, phone, birthday) VALUES ('Макарова', 'Анастасия', 'Глебовна', 2901348332, '89028381294', '1990.03.04')

INSERT INTO android_test.parents (id_parent, passport_mother, surname_mother, name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father, patronymic_father, phone_father, is_deptor) VALUES (5, '2910394873', 'Самодерженкова', 'Маргарита', 'Дмитриевна', '83404893851', '2920435204', 'Самодерженков', 'Дмитрий', 'Александрович', '86903058245', false)

INSERT INTO android_test.parents (id_parent, passport_mother, surname_mother, name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father, patronymic_father, phone_father, is_deptor) VALUES (6, '2901923848', 'Рыбкина', 'Надежда', 'Михайловна', '89303828438', '2920484842', 'Рыбкин', 'Кирилл', 'Романович', '89509888311', false)

INSERT INTO android_test.parents (id_parent, passport_mother, surname_mother, name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father, patronymic_father, phone_father, is_deptor) VALUES (7, '2901937374', 'Гурова', 'Ирина', 'Андреевна', '89202938421', '2900484888', 'Гуров', 'Максим', 'Викторович', '89020384322', false)

INSERT INTO android_test.parents (id_parent, passport_mother, surname_mother, name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father, patronymic_father, phone_father, is_deptor) VALUES (8, '2009923921', 'Грамматчикова', 'Анастасия', 'Игоревна', '80283742877', '2920348567', 'Грамматчиков', 'Артём', 'Михайлович', '89018374773', false)

INSERT INTO android_test.kids (surname, name_kid, patronymic, birthday, age, is_sick) VALUES ('Спиридонов', 'Андрей', 'Егорович', '2016.06.23', 5, false)

INSERT INTO android_test.kids (surname, name_kid, patronymic, birthday, age, is_sick) VALUES ('Бабкин', 'Игорь', 'Иванович', '2017.08.11', 4, false)

INSERT INTO android_test.kids (surname, name_kid, patronymic, birthday, age, is_sick) VALUES ('Алёшина', 'Василиса', 'Никитична', '2018.07.02', 3, false)

INSERT INTO android_test.kids (surname, name_kid, patronymic, birthday, age, is_sick) VALUES ('Беглецов', 'Артём', 'Максимович', '2018.11.26', 3, false)

INSERT INTO android_test.kids (surname, name_kid, patronymic, birthday, age, is_sick) VALUES ('Самодерженкова', 'Алёна', 'Дмитриевна', '2017.12.08', 4, false)

INSERT INTO android_test.kids (surname, name_kid, patronymic, birthday, age, is_sick) VALUES ('Рыбкин', 'Олег', 'Кириллович', '2015.03.09', 6, false)

INSERT INTO android_test.kids (surname, name_kid, patronymic, birthday, age, is_sick) VALUES ('Гурова', 'Арина', 'Максимовна', '2015.09.21', 6, false)

INSERT INTO android_test.kids (surname, name_kid, patronymic, birthday, age, is_sick) VALUES ('Гурова', 'Екатерина', 'Максимовна', '2015.09.21', 6, false)

INSERT INTO android_test.kids (surname, name_kid, patronymic, birthday, age, is_sick) VALUES ('Грамматчикова', 'Людмила', 'Артёмовна', '2016.03.08', 5, false)

INSERT INTO android_test.kids_and_parents (kid_id, parent_id) VALUES (1, 1)

INSERT INTO android_test.kids_and_parents (kid_id, parent_id) VALUES (2, 2)

INSERT INTO android_test.kids_and_parents (kid_id, parent_id) VALUES (3, 3)

```
INSERT INTO android_test.kids_and_parents (kid_id, parent_id) VALUES (4, 4)

INSERT INTO android_test.kids_and_parents (kid_id, parent_id) VALUES (5, 5)

INSERT INTO android_test.kids_and_parents (kid_id, parent_id) VALUES (6, 6)

INSERT INTO android_test.kids_and_parents (kid_id, parent_id) VALUES (7, 7)

INSERT INTO android_test.kids_and_parents (kid_id, parent_id) VALUES (8, 7)

INSERT INTO android_test.kids_and_parents (kid_id, parent_id) VALUES (9, 8)


INSERT INTO android_test.groups_ (name_group, num_of_kid, age_range) VALUES ('Сказка', 12, '3-4')

INSERT INTO android_test.groups_ (name_group, num_of_kid, age_range) VALUES ('Лучики', 10, '4-5')

INSERT INTO android_test.groups_ (name_group, num_of_kid, age_range) VALUES ('Малинка', 11, '5-6')


INSERT INTO android_test.group_defining (kid_id, group_id) VALUES (1, 2)

INSERT INTO android_test.group_defining (kid_id, group_id) VALUES (2, 1)

INSERT INTO android_test.group_defining (kid_id, group_id) VALUES (3, 1)

INSERT INTO android_test.group_defining (kid_id, group_id) VALUES (4, 1)

INSERT INTO android_test.group_defining (kid_id, group_id) VALUES (5, 1)

INSERT INTO android_test.group_defining (kid_id, group_id) VALUES (6, 3)

INSERT INTO android_test.group_defining (kid_id, group_id) VALUES (7, 3)

INSERT INTO android_test.group_defining (kid_id, group_id) VALUES (8, 3)

INSERT INTO android_test.group_defining (kid_id, group_id) VALUES (9, 2)
```

Фрагмент кода приложения

Код из класса для работы с подключенной БД

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Controls;
using MySql.Data.MySqlClient;

namespace DB_Store
{
```

```
public class DB
{
   MySqlConnection connection;
   MySqlCommand cmd;
   MySqlDataAdapter da;
   DataTable dt;
   DataGrid dataGrid;
   int rowsCount;
   public DB(DataGrid _dataGrid)
   {
      dataGrid = _dataGrid;
   }
   public void SetConnection(MySqlConnection _connection)
   {
      connection = _connection;
   }
   public void CloseConnection()
   {
      connection.Close();
   }
   public DataTable FillTable(string table, string name_parent)
   {
      string id_parent = name_parent;
      if (table == "Parents")
      {
         cmd = new MySqlCommand("SELECT * FROM parents WHERE
id_parent="+id_parent, connection);
      }
      if (table == "Admin_i")
      {/
         cmd = new MySqlCommand("SELECT * FROM administration WHERE
id_administrator=" + id_parent[id_parent.Length-1].ToString(), connection);
      }
      if (table == "Teachers")
         cmd = new MySqlCommand("select
id_teacher,surname_teacher,name_teacher,patronymic_teacher,passport,phone,birthday,name_of_subject
from teachers join subjects s on teachers.id_subject = s.id_subject", connection);
      if (table == "Kids")
      {/
         cmd = new MySqlCommand("select
id_kid,surname,name_kid,patronymic,birthday,name_group from kids join kids_and_parents kap on
kids.id_kid = kap.kid_id and kap.parent_id = "+ id_parent+ " join group_defining gd on kids.id_kid =
gd.kid_id join groups_ g on gd.group_id = g.id_group", connection);
      }
      if (table == "Kids_n")
```

```
                cmd = new MySqlCommand("select name_kid from kids join kids_and_parents kap on
kids.id_kid = kap.kid_id and kap.parent_id = " + id_parent + " join group_defining gd on kids.id_kid =
gd.kid_id join groups_ g on gd.group_id = g.id_group", connection);
            if (table == "Rub")
                cmd = new MySqlCommand("select cost from recipe_of_payment where parent_id = "
+ id_parent + " and status_of_pay = false", connection);
            if (table == "Tutors")
                cmd = new MySqlCommand("select
id_tutor,surname_tutor,name_tutor,patronymic_tutor,passport,phone,birthday from tutors", connection);
            if (table == "Baby")
                cmd = new MySqlCommand("select
id_babysitter,surname_babysitter,name_babysitter,patronymic_babysitter,passport,phone,birthday from
babysitters", connection);
            if (table == "teacher_for_subject")
            {
                cmd = new MySqlCommand("select id_teacher,surname_teacher from teachers where
id_subject = " + id_parent, connection);
            }
            if (table == "tutors_for_subject")
            {
                cmd = new MySqlCommand("select id_tutor,surname_tutor from tutors ", connection);
            }
            if (table == "baby_for_subject")
                cmd = new MySqlCommand("select id_babysitter,surname_babysitter from babysitters
", connection);
            if (table.Contains("groups_adm"))
                cmd = new MySqlCommand("select * from groups_", connection);
            if (table == "get_room")
                cmd = new MySqlCommand("select room from groups_ where id_group = " +
id_parent, connection);
            if (table == "Group_room")
                 cmd = new MySqlCommand("select name_group, room from groups_ join
group_defining gd on groups_.id_group = gd.group_id and gd.kid_id = " + id_parent, connection);
            da = new MySqlDataAdapter(cmd);
            dt = new DataTable();
            da.Fill(dt);
            return dt;
        }
        public DataTable FillTable(string table, string name_parent, string kid)
        {
            string id_parent = name_parent;
            if (table.Contains("Schedule"))
            {cmd = new MySqlCommand("select group_id from group_defining join
kids_and_parents kap on group_defining.kid_id = kap.kid_id where parent_id = " + id_parent + " and
kap.kid_id = " + kid , connection);
                da = new MySqlDataAdapter(cmd);
```

```
                dt = new DataTable();
                da.Fill(dt);
                string id_group = dt.Rows[0][0].ToString();
                cmd = new MySqlCommand("select
lesson_id,start_time,end_time,name_of_subject,surname_teacher,name_teacher,patronymic_teacher,surna
me_tutor,name_tutor,patronymic_tutor,surname_babysitter,name_babysitter,patronymic_babysitter from
schedule join subjects s on schedule.subject_id = s.id_subject join teachers t on schedule.teacher_id =
t.id_teacher join tutors t2 on schedule.tutor_id = t2.id_tutor join babysitters b on schedule.babysitter_id =
b.id_babysitter join lesson_period lp on schedule.lesson_id = lp.id_lesson where day_of_week = " +
table[table.Length - 1].ToString() + " and group_id = " + id_group, connection);
            }
            if (table == "Select_another")
                cmd = new MySqlCommand("select id_kid from kids join kids_and_parents kap on
kids.id_kid = kap.kid_id and kap.parent_id = " + id_parent + " join group_defining gd on kids.id_kid =
gd.kid_id join groups_ g on gd.group_id = g.id_group and name_kid = '" + kid+"'", connection);
            if (table.Contains("schedule_adm"))
            {
                string id_group = kid;
                cmd = new MySqlCommand("select id_Schedule,
start_time,end_time,name_of_subject,surname_teacher,name_teacher,patronymic_teacher,surname_tutor,
name_tutor,patronymic_tutor,surname_babysitter,name_babysitter,patronymic_babysitter from schedule
join subjects s on schedule.subject_id = s.id_subject join teachers t on schedule.teacher_id = t.id_teacher
join tutors t2 on schedule.tutor_id = t2.id_tutor join babysitters b on schedule.babysitter_id =
b.id_babysitter join lesson_period lp on schedule.lesson_id = lp.id_lesson where day_of_week = " +
table[table.Length - 1].ToString() + " and group_id = " + id_group, connection); }
            if (table.Contains("Pay_list")){
                if (kid == "All"){
                    cmd = new MySqlCommand("select id_parent, passport_mother, surname_mother,
name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father,
patronymic_father, phone_father, surname_bookkeeper, name_bookkeeper,patronymic_bookkeeper, cost
from parents join recipe_of_payment rop on parents.id_parent = rop.parent_id join bookkeeping b on
rop.bookkeeper_id = b.id_bookkeeper", connection);
                }
                else{
                    cmd = new MySqlCommand("select id_parent, passport_mother, surname_mother,
name_mother, patronymic_mother, phone_mother, passport_father, surname_father, name_father,
patronymic_father, phone_father, surname_bookkeeper, name_bookkeeper,patronymic_bookkeeper, cost
from parents join recipe_of_payment rop on parents.id_parent = rop.parent_id join bookkeeping b on
rop.bookkeeper_id = b.id_bookkeeper where status_of_pay = false", connection);} }
            da = new MySqlDataAdapter(cmd);
            dt = new DataTable();
            da.Fill(dt);
            if (table.Contains("Schedule"))
            {
                for (int i = 0; i < dt.Rows.Count; i++)
                {
```

```
                    string n = dt.Rows[i][4].ToString() + " " + dt.Rows[i][5].ToString()[0].ToString() + ". " + dt.Rows[i][6].ToString()[0].ToString() + ".";
                    dt.Rows[i][4] = n;
                    n = dt.Rows[i][7].ToString() + " " + dt.Rows[i][8].ToString()[0].ToString() + ". " + dt.Rows[i][9].ToString()[0].ToString() + ".";
                    dt.Rows[i][7] = n;
                    n = dt.Rows[i][10].ToString() + " " + dt.Rows[i][11].ToString()[0].ToString() + ". " + dt.Rows[i][12].ToString()[0].ToString() + ".";
                    dt.Rows[i][10] = n;
                }
                dt.Columns.Remove("name_teacher");
                dt.Columns.Remove("patronymic_teacher");
                dt.Columns.Remove("name_tutor");
                dt.Columns.Remove("patronymic_tutor");
                dt.Columns.Remove("name_babysitter");
                dt.Columns.Remove("patronymic_babysitter");
            }
            if (table.Contains("schedule"))
            {
                for (int i = 0; i < dt.Rows.Count; i++)
                {
                    string n = dt.Rows[i][4].ToString() + " " + dt.Rows[i][5].ToString()[0].ToString() + ". " + dt.Rows[i][6].ToString()[0].ToString() + ".";
                    dt.Rows[i][4] = n;
                    n = dt.Rows[i][7].ToString() + " " + dt.Rows[i][8].ToString()[0].ToString() + ". " + dt.Rows[i][9].ToString()[0].ToString() + ".";
                    dt.Rows[i][7] = n;
                    n = dt.Rows[i][10].ToString() + " " + dt.Rows[i][11].ToString()[0].ToString() + ". " + dt.Rows[i][12].ToString()[0].ToString() + ".";
                    dt.Rows[i][10] = n;
                }
                dt.Columns.Remove("name_teacher");
                dt.Columns.Remove("patronymic_teacher");
                dt.Columns.Remove("name_tutor");
                dt.Columns.Remove("patronymic_tutor");
                dt.Columns.Remove("name_babysitter");
                dt.Columns.Remove("patronymic_babysitter");
            }
            if (table.Contains("Pay_list"))
            {
                for (int i = 0; i < dt.Rows.Count; i++)
                {
                    string n = dt.Rows[i][1].ToString() + " " + dt.Rows[i][2].ToString()[0].ToString() + ". " + dt.Rows[i][3].ToString()[0].ToString() + ".";
                    dt.Rows[i][1] = n;
```

```
                    n = dt.Rows[i][6].ToString() + " " + dt.Rows[i][7].ToString()[0].ToString() + ". " +
dt.Rows[i][8].ToString()[0].ToString() + ".";
                    dt.Rows[i][6] = n;
                    n = dt.Rows[i][10].ToString() + " " + dt.Rows[i][11].ToString()[0].ToString() + ". " +
dt.Rows[i][12].ToString()[0].ToString() + ".";
                    dt.Rows[i][10] = n;
                }
                dt.Columns.Remove("name_mother");
                dt.Columns.Remove("patronymic_mother");
                dt.Columns.Remove("name_father");
                dt.Columns.Remove("patronymic_father");
                dt.Columns.Remove("name_bookkeeper");
                dt.Columns.Remove("patronymic_bookkeeper");
            }
            return dt;
        }
        public void SelectTable(string table, string user_name)
        {
            dt = FillTable(table, user_name);
        }
```