МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Основная профессиональная образовательная программа
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль) «Технологии разработки программного обеспечения»
форма обучения – очная

ОТЧЁТ

по реализации проекта для дисциплины «Базы данных» по направлению “09.03.01 – Информатика и вычислительная техника” (профиль: “Технологии разработки программного обеспечения”)

Преподаватель: к.ф-м.н., доцент кафедры ИТиЭО

_____
(Жуков Н. Н.)

Преподаватель: ассистент кафедры ИТиЭО

_____
(Иванова Е. А.)

Студенты 2 курса:

Сорокина И. И. _____
Беленко А. В. _____
Чалапко Е. В. _____

Санкт-Петербург
2021

Оглавление

Ответственные

Сорокина И. И. – разработчик проекта. В обязанности Сорокиной И. И. входил процесс нормализации базы данных и её реализации, руководительство проекта. При выполнении данного задания были использованы знания по следующим формам нормализации: 1НФ-3НФ.

Беленко А. В. – разработчик проекта. В обязанности Беленко А. В. входил процесс нормализации базы данных и её реализации, разработка плана работы над проектом. При выполнении данного задания были использованы знания по следующим формам нормализации: 1НФ-3НФ.

Чалапко Е. В. – разработчик проекта. В обязанности Чалапко Е. В. входил процесс нормализации базы данных и её реализации, сбор необходимой информации для проекта. При выполнении данного задания были использованы знания по следующим формам нормализации: 1НФ-3НФ.

Предметная область

Перед разработчиками была поставлена задача разработки варианта более удобного предоставления домашнего задания, и расписания занятий в институте. Для этого им был предоставлен доступ к центру дистанционной поддержки обучения РГПУ им. А. И. Герцена, электронному расписанию студента и предстоящим заданиям. Для выполнения поставленной задачи, разработчикам необходимо создать Telegram бота который сможет по запросу выдавать студенту задания и расписание, для подгруппы этого студента.

По мере увеличения объема данных, потребовалось организовать их в виде базы данных:

- спроектировать реляционную базу данных (был выявлен список сущностей, список атрибутов сущностей и проанализированы взаимосвязи между сущностями);
- провести нормализацию полученных сущностей;
- представить ER-диаграмму;

Ход выполнения нормализации

1. Процедуры концептуального проектирования

1.1 Определение сущностей и их документирование

- Сущность "Студент/student":

этот объект будет описывать пользователей базы данных.

Атрибуты: уникальный номер, фамилия, имя, отчество, номер группы(основная).

- Сущность "Группа/group":

это объект будет хранить список групп, для которых существуют задания.

Атрибуты: номер, наименование, администратор.

- Сущность "Расписание/schedule":

это объект хранит расписание занятий на семестр для конкретной группы.

Атрибуты: дата первого занятия, дата последнего занятия, время начала занятия, время окончания занятия, верхняя неделя, нижняя неделя, номер дисциплины.

- Сущность "Дисциплины/subjects":

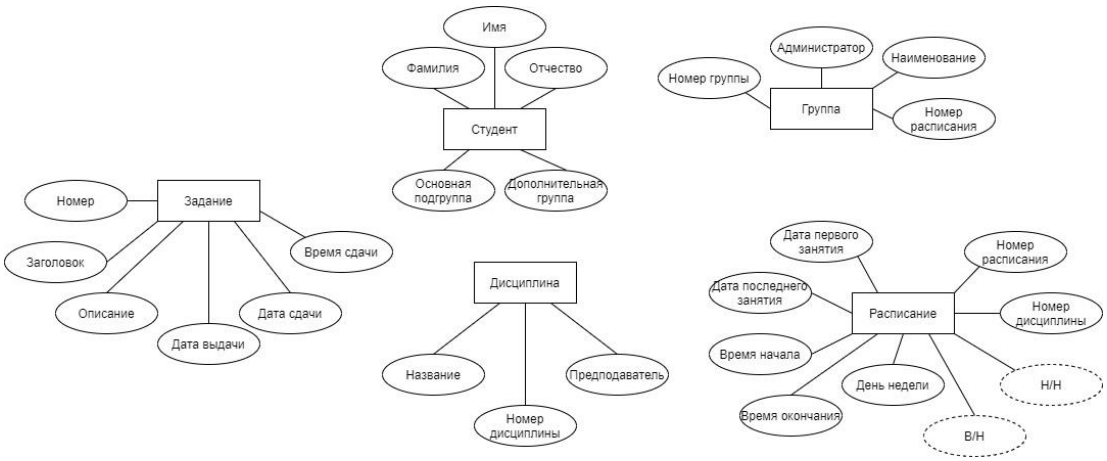
этот объект хранит информацию о изучаемой дисциплине.

Атрибуты: номер дисциплины, название, преподаватель.

Сущность "Задания/tasks":

это объект хранит информацию о всех заданиях, полученных студентами в текущем семестре.

Атрибуты: номер задания, заголовок, описание, дата выдачи, дата сдачи, время сдачи.



1.2 Определение связей между сущностями и их документирование

- Связь "студент-группа"(1:1):

обязательная, один студент относится к одной группе;

- Связь "группа-расписание"(1:N):

обязательная, одна группа содержит n-записей в таблице расписание;

- Связь "расписание-группа"(N:1):

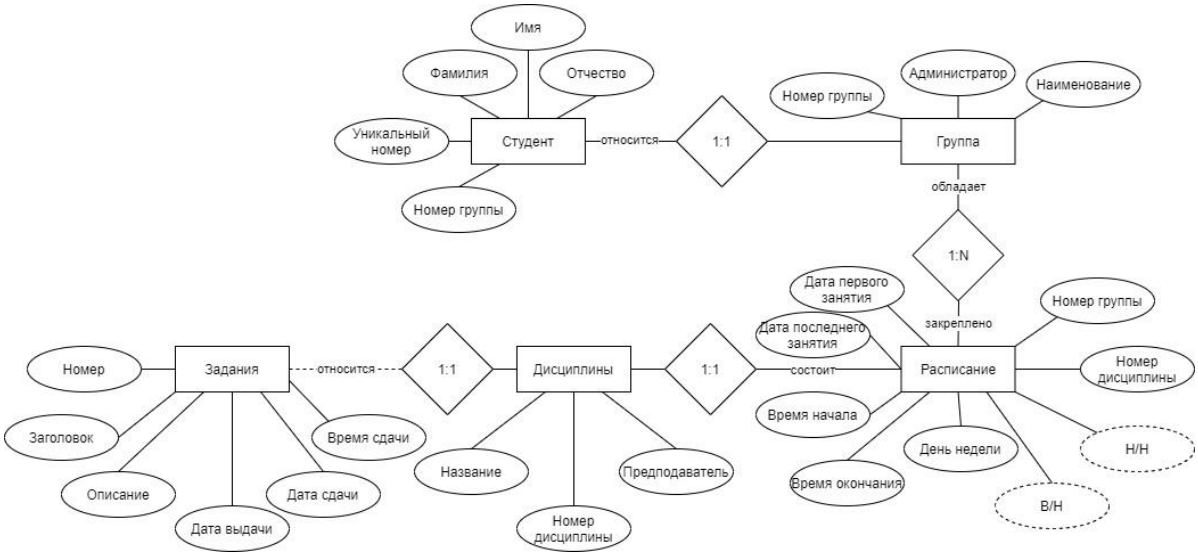
обязательная, n-записей из таблицы расписание принадлежат одной группе;

- Связь "расписание-дисциплины"(1:1):

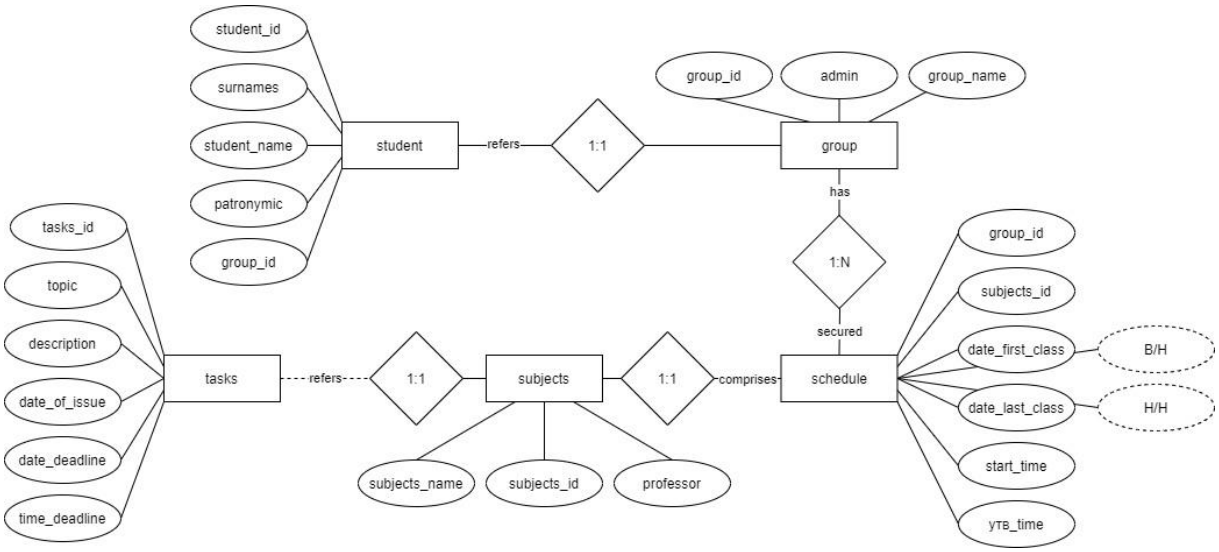
обязательная, одна запись в таблице расписание относится к одной дисциплине;

- Связь "задания-дисциплины"(1:1):

необязательная, одно задание относится к одной дисциплине.



1.3 Создание ER-модели предметной области



1.4 Определение атрибутов, их значений и ключей для сущностей.

Выделенный жирным атрибут - **первичный ключ**

Имя атрибута	Описание	Тип данных	Размер	Default	NULL	Простой / Составной	Расчетный	ОДЗ
student								
student_id	Уникальный номер студента	INT	-	-	NOT NULL	Простой	-	
surnames	Фамилия	VARCHAR	100	-	NOT NULL	Простой	-	Не содержит цифр
student_name	Имя	VARCHAR	50	-	NOT NULL	Простой	-	Не содержит цифр
patronymic	Отчество	VARCHAR	100	-		Простой	-	Не содержит цифр
group_id	Уникальный номер группы	INT	-	-	NOT NULL	Простой	-	
group								

group_id	Уникальный номер группы	INT	-	-	NOT NULL	Простой	-	
admin	id администратора группы	INT	-	-	NOT NULL	Простой	-	
group_name	Удобочитаемое наименование: ИТВ(1)/19	VARCHAR	50	-	NOT NULL	Простой	-	
schedule								
group_id	Уникальный номер группы	INT	-	-	NOT NULL	Простой	-	
subjects_id	Уникальный номер дисцип.	INT	-	-	NOT NULL	Простой	-	
date_first_class	Дата первого занятия	DATE	-	-	NOT NULL	Простой	-	Даты в диапазоне текущего семестра
date_last_class	Дата последнего занятия	DATE	-	-	NOT NULL	Простой	-	Даты в диапазоне текущего семестра
start_time	Время начала занятий	TIME	-	-	NOT NULL	Простой	-	с 8:00 до 21:00
end_time	Время окончания занятий	TIME	-	-	NOT NULL	Простой	-	с 8:00 до 21:00
B_H	верхняя неделя	BOOLEAN	-	TRUE	NOT NULL	Простой	-	

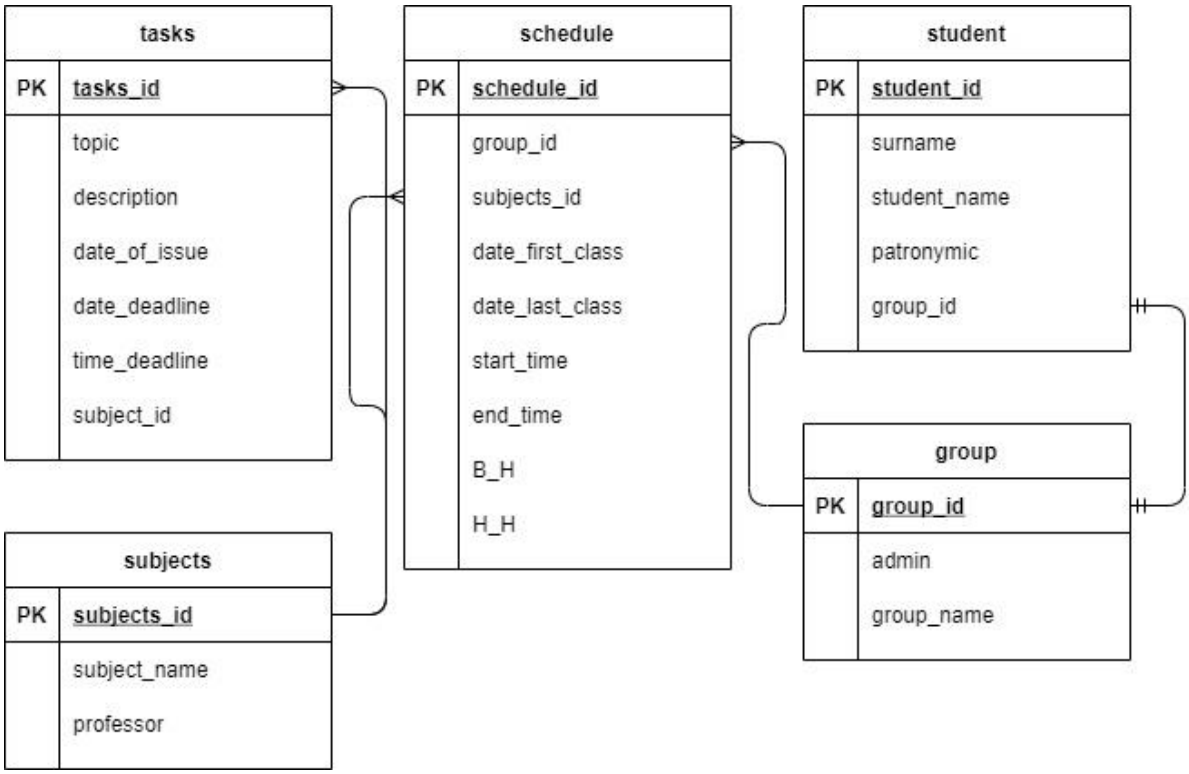
H_H	нижняя неделя	BOOLEAN	-	TRUE	NOT NULL	Простой	-	
schedule_id	Уникальный идентификатор	INT	-		NOT NULL	Простой	-	
subjects								
subjects_name	Название	VARCHAR	200	-	NOT NULL	Простой	-	
subjects_id	Уникальный номер дисципл.	INT	-	-	NOT NULL	Простой	-	
professor	Преподаватель	VARCHAR	100	-	NOT NULL	Простой	-	Не содержит цифр
tasks								
tasks_id	Номер задачи	INT	-	-	NOT NULL	Простой	-	
topic	Заголовок	VARCHAR	100	-	NOT NULL	Простой	-	
description	Описание	TEXT	-	-		Простой	-	
date_of_issue	Дата выдачи	DATE	-	-		Простой	-	Даты в диапазоне текущего семестра
date_deadline	Дата сдачи	DATE	-	-		Простой	-	Даты в диапазоне текущего семестра
time_deadline	Время сдачи	TIME	-	-		Простой	-	

2. Процедуры логического проектирования

2.1 Выбор модели данных

Была выбрана **реляционная модель данных** в связи с наглядностью табличного представления данных и удобства работы с ними.

2.2 Определение набора таблиц исходя из ER-модели и их документирование



2.3 Нормализация таблиц

1НФ:

Таблица находится в 1НФ, если все её поля содержат только простые не делимые значения. Данное требование выполняется для всех таблиц.

2НФ:

Таблица находится в 2НФ, если она удовлетворяет требованиям 1НФ и не ключевые поля функционально полно зависят от первичного ключа.

Изменениям подверглась таблица schedule, был удален столбец schedule_id, вместо него в качестве первичного ключа используется составной ключ group_id + subject_id. Все остальные таблицы соответствуют требованиям.

3НФ:

Таблица находится в 3НФ если она удовлетворяет требованиям 2НФ и не содержит транзитивных зависимостей. Транзитивная зависимость - зависимость между не ключевыми полями. Транзитивные зависимости не выявлены не в одной из таблиц.

НФБК:

Таблица находится в нормальной форме Бойса — Кодда тогда и только тогда, когда детерминанты всех её функциональных зависимостей являются потенциальными ключами. Дополнительных зависимостей в структурах таблиц не найдено.

4НФ:

Таблица находится в 4НФ, если она находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от её потенциальных ключей. Была обнаружена зависимость внутри таблицы schedule, для её устранения была создана дополнительная таблица group_schedule в которую будут заноситься данные о том, к какой группе относится строка записей в таблице с расписанием, таким образом мы избавились от тривиальной многозначной зависимости и достигли 4НФ,

5НФ:

Таблица находится в пятой нормальной форме тогда и только тогда, когда она находится в 4НФ и каждая нетривиальная зависимость определяется ее потенциальным ключом. Нетривиальные зависимости не обнаружены, значит отношения приведены к 5НФ.

2.4 Проверка логистической модели данных на предмет возможности выполнения всех транзакций, предусмотренных пользователями.

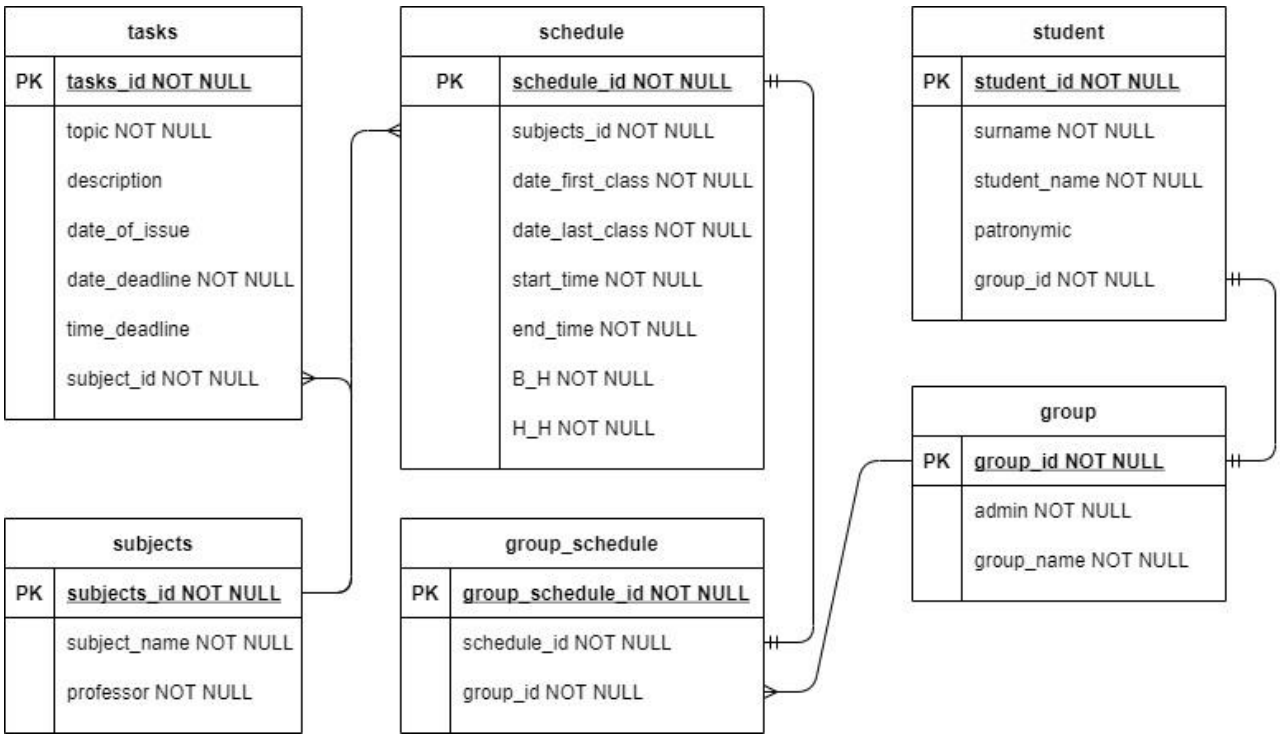
Наша база данных должна по номеру студента определить расписание и список заданий. Для получения расписания, мы берем индивидуальный номер студента и по нему определяем номер группы, по номеру группы мы получаем список записей в таблице с расписанием, которые относятся к интересующей нас группе, форматируем его и выводим расписание на требуемые даты, для получения заданий, берем список из расписания и вытягиваем из него номера дисциплин, по номерам дисциплин выводим все имеющиеся задания. Существующая модель базы данных, отвечает поставленным требованиям.

2.5 Определением требований поддержки целостности данных и их документирование

Обязательные данные (атрибуты, которые не могут иметь NULL значения) отмечены подписью 'Not null' на общей модели данных. Все первичные ключи и внешние ключи не могут содержать значения null, что обеспечивает целостность сущностей и ссылок. Ограничения значения атрибутов:

- **Таблица tasks** - дата выдачи задания не может быть больше чем дата сдачи.
- **Таблица subjects** - атрибут профессор может содержать только буквенные значения.
- **Таблица scheduke** - дата начала занятий не может быть больше чем дата окончания занятий, время начала занятий не может быть больше чем время окончания занятий.
- **Таблица student** - фамилия, имя и отчество могут содержать только буквенные значения.

Итоговая логическая модель данных:



3.Процедуры физического проектирования

3.1 Проектирование таблиц базы данных средствами выбранной СУБД

Для реализации была выбрана SQLite, так как это изумительная библиотека, встраиваемая в приложение, которое её использует. Будучи файловой БД, она предоставляет отличный набор инструментов для более простой (в сравнении с серверными БД) обработки любых видов данных.

Запросы на создание таблиц

```
CREATE TABLE "subjects" (
	"subjects_id"	INTEGER NOT NULL UNIQUE,
	"subjects_name"	TEXT NOT NULL,
	"professor"	TEXT NOT NULL,
	PRIMARY KEY("subjects_id" AUTOINCREMENT)
);

CREATE TABLE "group" (
	"group_id"	INTEGER NOT NULL UNIQUE,
	"admin"	INTEGER NOT NULL,
	"group_name"	TEXT NOT NULL,
	PRIMARY KEY("group_id" AUTOINCREMENT)
);

CREATE TABLE "student" (
	"student_id"	INTEGER NOT NULL UNIQUE COLLATE UTF16CI,
	"surname"	TEXT NOT NULL COLLATE UTF16CI,
	"student_name"	INTEGER NOT NULL,
	"patronymic"	TEXT,
	"group_id"	INTEGER NOT NULL COLLATE UTF16CI,
	PRIMARY KEY("student_id"),
	FOREIGN KEY("group_id") REFERENCES "group"("group_id")
);
```

```
CREATE TABLE "schedule" (
	"schedule_id" INTEGER NOT NULL UNIQUE,
	"subjects_id"  INTEGER NOT NULL COLLATE UTF16CI,
	"date_first_class"	TEXT NOT NULL,
	"date_last_class"	NUMERIC NOT NULL,
	"start_time"	TEXT NOT NULL,
	"end_time"	TEXT NOT NULL,
	"B_H" INTEGER NOT NULL DEFAULT 1,
	"H_H" INTEGER NOT NULL DEFAULT 1,
	PRIMARY KEY("schedule_id" AUTOINCREMENT),
	FOREIGN KEY("subjects_id") REFERENCES "subjects"("subjects_id")
);

CREATE TABLE "group_schedule" (
	"group_schedule_id"  INTEGER NOT NULL UNIQUE,
	"schedule_id" INTEGER NOT NULL,
	"group_id"	INTEGER NOT NULL,
	FOREIGN KEY("schedule_id") REFERENCES "schedule"("schedule_id"),
	FOREIGN KEY("group_id") REFERENCES "group"("group_id"),
	PRIMARY KEY("group_schedule_id" AUTOINCREMENT)
);

CREATE TABLE "tasks" (
	"task_id"	INTEGER NOT NULL UNIQUE,
	"topic" TEXT NOT NULL,
	"description"	TEXT,
	"date_of_issue"	TEXT,
```

```
    "date_deadline"         TEXT NOT NULL,
    "time_deadline"         TEXT,
    "subject_id"    INTEGER NOT NULL,
    FOREIGN KEY("subject_id") REFERENCES "subjects"("subjects_id"),
    PRIMARY KEY("task_id" AUTOINCREMENT)
);
```

Объяснение выбранной СУБД

Для данного проекта была выбрана СУБД SQLite. Такой выбор был произведён из-за следующих особенностей СУБД:

- Файловая структура - вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины
- Отсутствие необходимости настройки сервера СУБД
- Полностью свободная лицензия
- Кроссплатформенность
- Высокая скорость простых операций выборки данных
- Безопасность. БД хранится в одном файле, права доступа к которому можно контролировать стандартными средствами ОС
- Очень экономичная, в плане ресурсов, архитектура.
- Простота использования.

Функционал бота

- **/start** - запуск, если пользователь есть в базе данных, то приветствие по имени, если нет то предложение познакомится, переход к /new;
- **/new** - запись нового пользователя в базу данных;
- **/schedule** - вывод расписания (будет дорабатываться, просто сырой вывод расписания по номеру группы);
- **/tasks** - вывод заданий (будет дорабатываться, просто сырой вывод заданий по номеру группы).

Направления дальнейшей разработки

Планируется внедрить:

- Вывод расписания на сегодня/завтра/всю неделю по кнопкам;
- Вывод заданий по конкретной дисциплине/на конкретный период;
- Создание прав администратора для добавления заданий и внесения изменений.

Процесс разработки

Для разработки использовались:

- **Библиотека AIOGram** - довольно простой и полностью асинхронный фреймворк для Telegram Bot API, написанный на Python 3.7. Позволяет создавать ботов быстрее и проще.
- **IDE - PyCharm**
- **Встроенная библиотека SQLite3**
- **DB Browser (SQLite)** - программа для взаимодействия с БД
- **Github** и **Heroku** - для деплоя и запуска бота

Изначально необходимо было получить бот токен, это делается через специального бота в телеграмме, затем при помощи библиотеки **AIOGram** была создана заготовка для бота, основной функционал в файле **handlers.py**, там расположены описания хэндлеров.

Хендлеры - это декораторы, которые при помощи шаблонов, отлавливают сообщения пользователей и запускают, в зависимости от содержимого, различные функции.

Пользователь что-то пишет, сообщение "пролетает" через все хенделеры и если подходит куда-то по фильтрам то запускает функцию, таких хендлеров может быть сколько угодно.

В процессе знакомства бота с новым пользователем, реализована машина состояний (Finite State Machine/ Конечный автомат. Конечный автомат (или попросту FSM — Finite-state machine) это модель вычислений, основанная на гипотетической машине состояний. В один момент времени только одно состояние может быть активным. Следовательно, для выполнения каких-либо действий машина должна менять свое состояние.

Пример кода

Примечание: код, показанный здесь может подвергнутся изменениям

-файл **database.py**

```python
import sqlite3
conn = sqlite3.connect('project.db')
cur = conn.cursor()


def test():
    cur.execute("SELECT * FROM student;")
    all_results = cur.fetchall()
    return all_results


def search_id(user_id):
    cur.execute(f'''SELECT student_name FROM student
                    WHERE student_id = {user_id};''')
    one_result = cur.fetchone()
    return one_result


def writing_student(user_list):
    cur.execute('''INSERT INTO student(student_id, surname, student_name, group_id)
                    VALUES(?, ?, ?, ?);''', user_list)
    conn.commit()


# запрос на выдачу расписания по номеру группы
def give_schedule(id_group):
    cur.execute(f'''SELECT schedule.start_time, schedule.end_time, subjects.subjects_name
                FROM group_schedule, schedule, subjects
                WHERE  group_schedule.group_id = {id_group}
                AND group_schedule.schedule_id = schedule.schedule_id
                AND schedule.subjects_id = subjects.subjects_id
                ;''')
    all_results = cur.fetchall()
    return all_results


# запрос для выдачи заданий по номеру группы (создала новую таблицу)
def give_tasks(id_group):
    cur.execute(f'''SELECT  subjects.subjects_name, topic, description, date_deadline, time_deadline
                FROM tasks, subjects, tasks_for_group
                WHERE tasks.subject_id = subjects.subjects_id
                AND tasks_for_group.task_id = tasks.task_id
                AND tasks_for_group.group_id = {id_group}
                ;''')
    all_results = cur.fetchall()
    return all_results
```

-примеры из файла **handlers.py**

```python
# запуск
@dp.message_handler(commands=['start'])
async def process_start_command(message: types.Message):
    # проверка id пользователя в базе данных
    user = search_id(message.chat.id)
    if user is not None:
        await message.answer(f"Привет, {user[0]}!\n")
    else:
        await message.answer(f"Привет!\nЯ бот-помощник!\nДавай знакомиться!\nЖми /new")


# знакомство
@dp.message_handler(Command("new"), state=None)
async def new_user(message: types.Message):
    await message.answer("Что бы я мог помогать, мне нужно узнать тебя:\n"
                         "Вопрос №1. \n\n"
                         "Имя:")
    await Test.Q1.set()
```

```python
# выдача расписания
@dp.message_handler(commands=['schedule'])
async def echo(message: types.Message):
    id_group = student_group(message.chat.id)
    text = give_schedule(id_group[0])
    await message.answer(text=text)


# выдача заданий
@dp.message_handler(commands=['tasks'])
async def echo(message: types.Message):
    id_group = student_group(message.chat.id)
    text = give_tasks(id_group[0])
    await message.answer(text=text)


# help
@dp.message_handler(commands=['help'])
async def echo(message: types.Message):
    text = ''' /schedule - расписание твоей группы\n/tasks - задания твоей группы'''
    await message.answer(text=text)
```

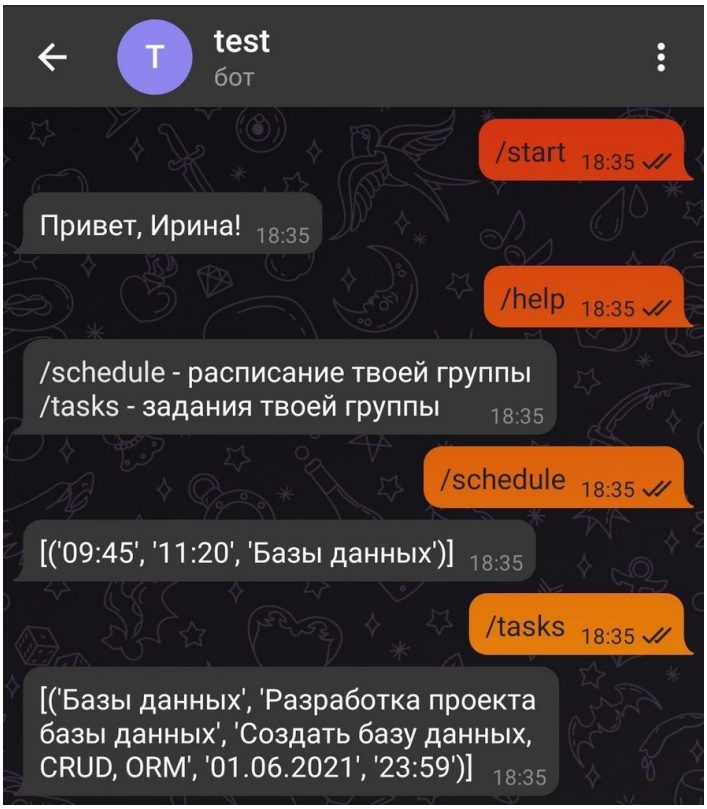
-пример меню из файла **menu.py**

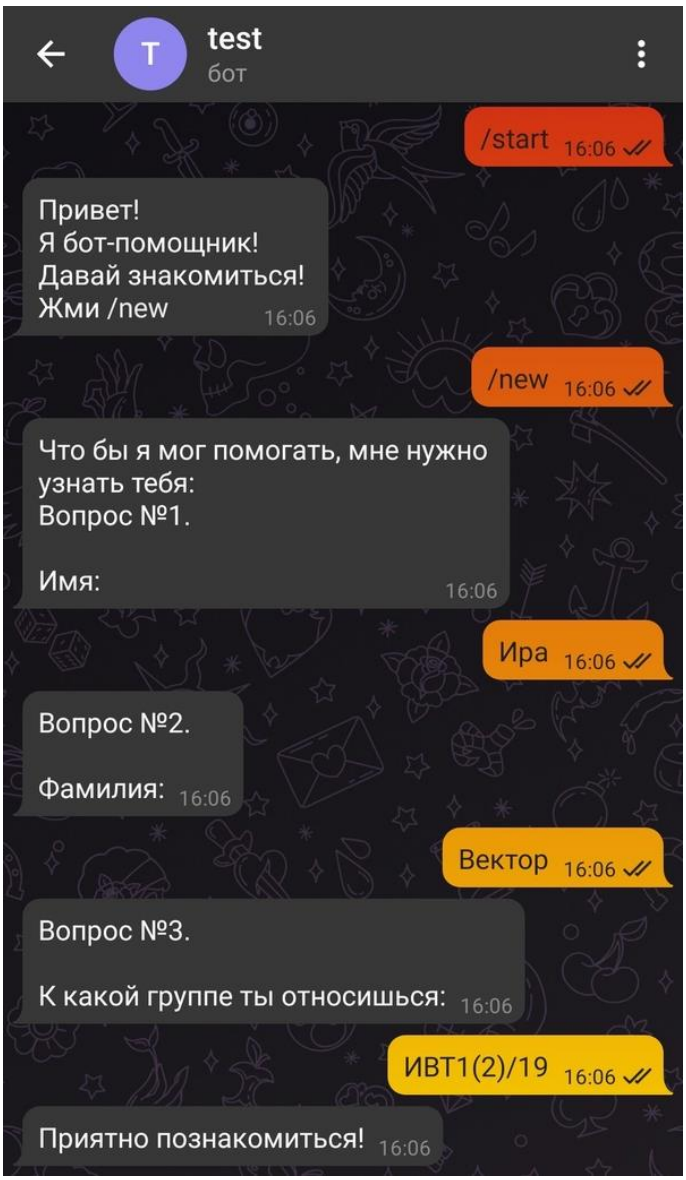
```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

menu = ReplyKeyboardMarkup(
    keyboard=[
        [
            KeyboardButton(text='ИВТ1(1)/19'),
            KeyboardButton(text='ИВТ1(2)/19'),
            KeyboardButton(text='ИВТ2/19')
        ]
    ],
    resize_keyboard=True
)
```

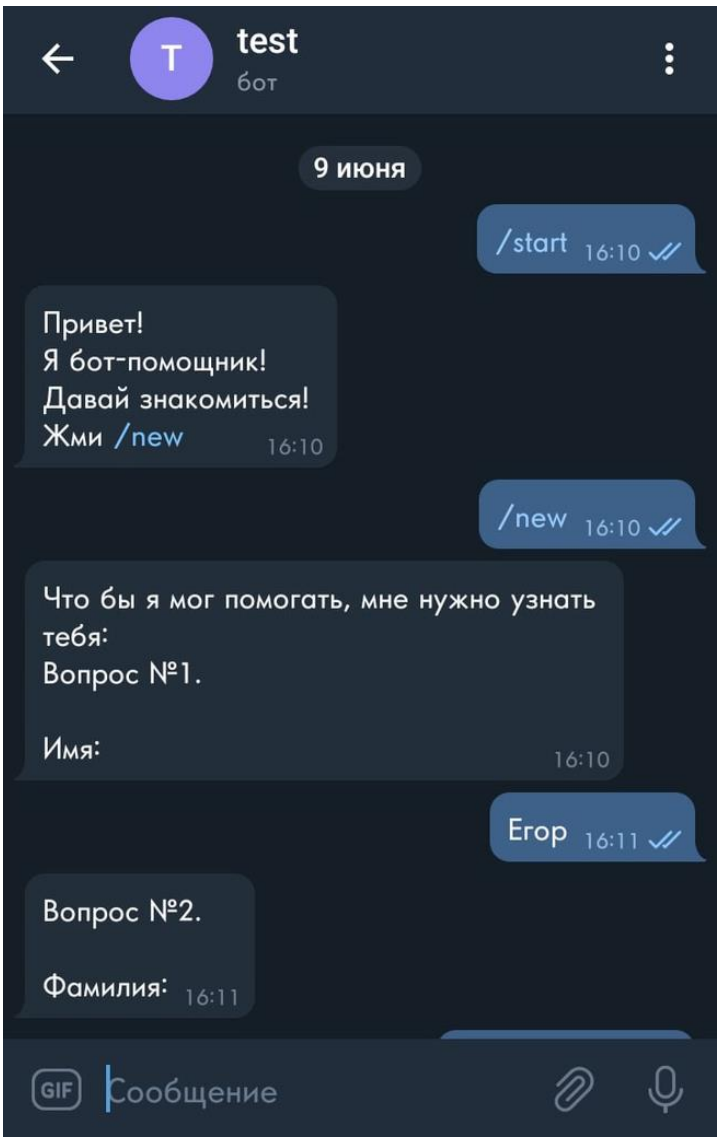
-пример варианта основного файла из файла **main.py**

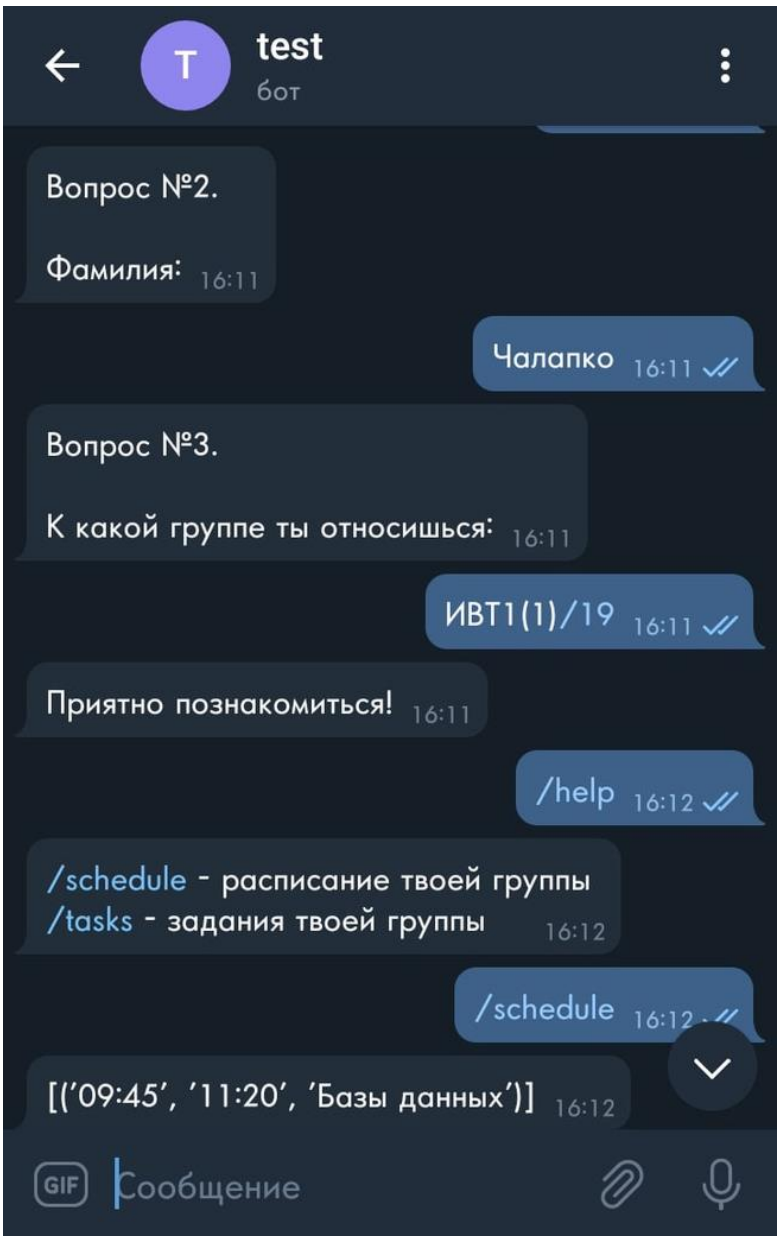
```
import asyncio, logging

from aiogram import Bot, Dispatcher, executor, types
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from config import BOT_TOKEN

loop = asyncio.get_event_loop()
bot = Bot(BOT_TOKEN, parse_mode=types.ParseMode.HTML)
storage = MemoryStorage()
dp = Dispatcher(bot, loop=loop, storage=storage)

logging.basicConfig(format=u'%(filename)s [LINE:%(lineno)d] #%(levelname)-8s [%(asctime)s]  %(message)s',
                    level=logging.INFO,
                    )


async def on_shutdown(dp):
    await bot.close()
    await storage.close()

if __name__ == '__main__':
    from handlers import dp, send_to_admin
    executor.start_polling(dp, on_startup=send_to_admin, on_shutdown=on_shutdown)
```

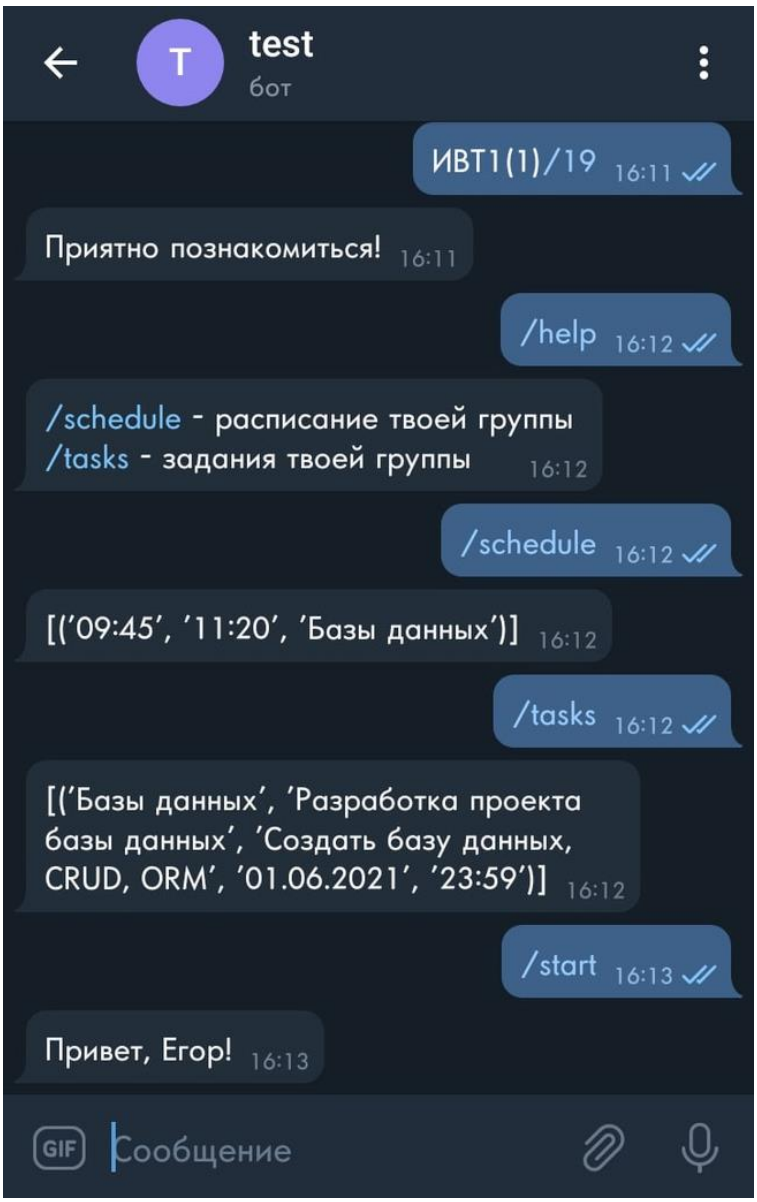
Примеры работы бота

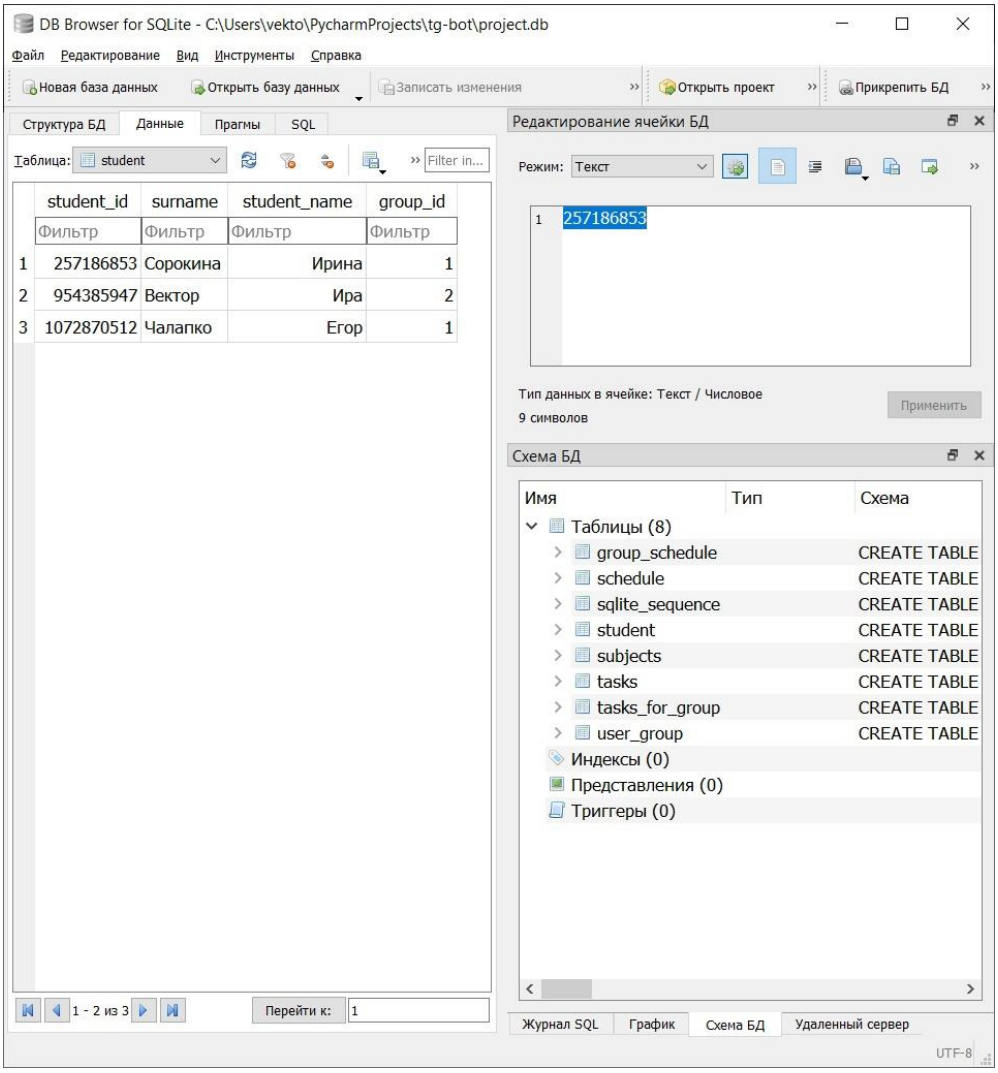



test
бот
/start 16:06
Привет!
Я бот-помощник!
Давай знакомиться!
Жми /new 16:06
/new 16:06
Что бы я мог помогать, мне нужно узнать тебя:
Вопрос №1.
Имя: 16:06
Ира 16:06
Вопрос №2.
Фамилия: 16:06
Вектор 16:06
Вопрос №3.
К какой группе ты относишься: 16:06
ИВТ1(2)/19 16:06
Приятно познакомиться! 16:06


test
бот
9 июня
/start 16:10
Привет!
Я бот-помощник!
Давай знакомиться!
Жми /new 16:10
/new 16:10
Что бы я мог помогать, мне нужно узнать тебя:
Вопрос №1.
Имя: 16:10
Егор 16:11
Вопрос №2.
Фамилия: 16:11
GIF
Сообщение


test
бот
Вопрос №2.
Фамилия: 16:11
Чалапко 16:11
Вопрос №3.
К какой группе ты относишься: 16:11
ИВТ1(1)/19 16:11
Приятно познакомиться! 16:11
/help 16:12
/schedule - расписание твоей группы
/tasks - задания твоей группы 16:12
/schedule 16:12
[('09:45', '11:20', 'Базы данных')] 16:12
GIF
Сообщение


test
бот
ИВТ1(1)/19 16:11
Приятно познакомиться! 16:11
/help 16:12
/schedule - расписание твоей группы
/tasks - задания твоей группы 16:12
/schedule 16:12
[('09:45', '11:20', 'Базы данных')] 16:12
/tasks 16:12
[('Базы данных', 'Разработка проекта базы данных', 'Создать базу данных, CRUD, ORM', '01.06.2021', '23:59')] 16:12
/start 16:13
Привет, Егор! 16:13
GIF
Сообщение


DB Browser for SQLite - C:\Users\vekto\PycharmProjects\tg-bot\project.db
Файл Редактирование Вид Инструменты Справка
Новая база данных
Открыть базу данных
Записать изменения
Открыть проект
Прикрепить БД
Структура БД
Данные
Прагмы
SQL
Таблица: student
Filter in...
student_id surname student_name group_id
Фильтр Фильтр Фильтр Фильтр
1 257186853 Сорокина Ирина 1
2 954385947 Вектор Ира 2
3 1072870512 Чалапко Егор 1
1 - 2 из 3
Перейти к: 1
Редактирование ячейки БД
Режим: Текст
1 257186853
Тип данных в ячейке: Текст / Числовое
9 символов
Применить
Схема БД
Имя Тип Схема
Таблицы (8)
group_schedule CREATE TABLE
schedule CREATE TABLE
sqlite_sequence CREATE TABLE
student CREATE TABLE
subjects CREATE TABLE
tasks CREATE TABLE
tasks_for_group CREATE TABLE
user_group CREATE TABLE
Индексы (0)
Представления (0)
Триггеры (0)
Журнал SQL
График
Схема БД
Удаленный сервер
UTF-8