Управление образования города Пензы

МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ГИМНАЗИЯ № 44 ГОРОДА ПЕНЗЫ

***Творческий проект***

**«Тестирующая система для школьников в мессенджере Telegram»**

Учащийся Россеев Илья Викторович

10 класс

Руководитель Сафронкин Николай Юрьевич

Учитель информатики

Пенза 2023

[1. Глава 1. Подготовительный этап 9](#_Toc128922327)

[1.1 Что такое Telegram (Телеграм)? 9](#_Toc128922328)

[1.2 Что такое телеграм-бот? 10](#_Toc128922329)

[1.3 Что такое API? 12](#_Toc128922330)

[1.4 VCS (Version Control Systems) 14](#_Toc128922331)

[1.5 Базы данных (SQLite). 16](#_Toc128922332)

[1.6 Python. 18](#_Toc128922333)

[1.7 Библиотеки. 20](#_Toc128922334)

[2. Глава 2. Разработка программного кода 21](#_Toc128922335)

[2.1 main.py 21](#_Toc128922336)

[2.2 to\_result\_db.py, to\_telegram\_db.py 21](#_Toc128922337)

[2.3 config.py 21](#_Toc128922338)

[2.4 B\_D 21](#_Toc128922339)

[2.5 Base 22](#_Toc128922340)

[2.6 Устройство баз данных 22](#_Toc128922341)

[3. Глава 3. Устройство чат бота 24](#_Toc128922342)

[3.1 Окно регистрации 24](#_Toc128922343)

[3.2 Основное окно 25](#_Toc128922345)

[3.3 Окно помощи 26](#_Toc128922347)

[3.4 Окно выбора варианта 27](#_Toc128922349)

[3.5 Пример решения варианта 28](#_Toc128922351)

**Введение**

Каждый человек в течение своей жизни постоянно учится. В детстве, с первых дней нашей жизни, мы учимся познавать этот мир. Наблюдаем за окружающей действительностью. Мы учимся ходить и говорить. Мы осознаем себя как частичку огромного мира. Первые знания передают нам наши родители, наши мамы и папы. А затем их дело продолжает школа и учителя.

Работа учителя не заканчивается, когда он покидает здание школы после уроков, он должен все время совершенствоваться и расти, только так он сможет воспитать в себе высокие моральные качества и научить тому же детей. Учитель учит детей и оценивает их знания в совершенно различных областях.

Мой проект нацелен на упрощение учебного процесса, как для учеников, так и для учителей. Я хочу, чтобы учителя тратили меньше времени на проверку на составление, сбор, проверку самостоятельных/домашних/контрольный работ школьников. А ученики могли выполнять их в удобном для них приложении.

Тем самым моё желание – облегчить жизнь ученикам и учителям и стало основанием для формулировки **проблемы:** как помочь учителям с проверкой заданий школьников?

**Тема** моего проекта: «Тестирующая система для школьников в мессенджере Telegram».

**Цель** - создать чат бота в мессенджере «Telegram», т.к. он насчитывает уже более миллиарда скачиваний, и охватывает очень большую аудиторию среди молодёжи.

В соответствии с проблемой и целью проекта были поставлены следующие задачи**:**

1. анализ основных используемых материалов и пособий
2. построение чёткого плана работы
3. разработка рабочего программного кода
4. тестирование работоспособности бота в локальной среде
5. тестирование работоспособности бота на школьном сервере

**Актуальность** заключается в том, что при работе над программным кодом у меня будет возможность углублённо изучить язык программирования «**Python**», а также несколько библиотек, которые используются в промышленном программировании (Telegram Bot API, sqlite3), познакомиться с системой контроля версий (Version Control Systems или VCS) – Git, программой SQLiteStudio. Это даст возможность мне сформировать практическое представление о программировании на языке «**Python**», что в последующем поможет при сдаче ЕГЭ и поступлении в ВУЗ на специальность программиста.

В перспективе, при положительном результате данного проекта, я хотел бы продолжить заниматься программированием и постепенно решать более сложные задачи.

**Новизна** заключается в том, что я хотел бы дать возможность пользования ботом учителям и ученикам Гимназии №44 г. Пензы на добровольной основе, что положительно бы сказалось на рейтинге школы в индустрии информационных технологий.

# Глава 1. Подготовительный этап

1. Что такое Telegram (Телеграм)?

***Telegram (Телеграм)***

Телеграм — это мессенджер (программа для обмена сообщениями), реализованный по клиент-серверной архитектуре. Используя сервер для создания диалога между двумя клиентами, Телеграм пересылает через него или напрямую текстовые сообщения, а также изображения, видео или документы других форматов.

1. Что такое ****телеграм-бот?****

Представьте, что у нас есть зарегистрированный аккаунт. Все сообщения, присылаемые на него, попадают на вход некоторой программе, а выход этой программы отправляется в виде сообщения тому, от кого пришло исходное сообщение. Это и есть бот. Иными словами, Телеграм-бот — это специальный пользователь, поведением которого управляет некоторая программа. Технически для сервера нет разницы, является данный пользователь человеком или ботом: для сервера оба клиента выглядят одинаково. Эта идеология очень похожа на пользователей-сообществ в vk.com.

API, которое я изучил, позволяет удобным образом получать сообщения от пользователей, обрабатывать их и отсылать ответы. Вся системная часть (получение ключей, шифрование, маршрутизация и т. д.) остается зоной ответственности библиотеки и архитектуры Telegram. Разработчику же дается удобный интерфейс, поэтому создателю нужно только реализовать логику поведения бота.

Какие задачи можно решать Телеграм-ботом?

Все ограничивается только фантазией. Например:

* Автоответчики. Все ситуации, когда требуется однозначный ответ на запрос. Например, бот может сообщать телефоны и другие контакты организации, ее рабочее время или предоставлять другую справочную информацию по запросу
* Интерфейс доступа к веб-сервисам. Бот может выполнять запросы к различным API и отдавать ответы в виде телеграм-сообщений
* Сценарии действий. Бот может пройти по какому-либо сценарию, задать пользователю определенные вопросы и собрать ответы на них. Например, при регистрации в каком-либо сервисе или при заявке на услугу
* Игры. Бот умеет пересылать картинки, поэтому можно создавать любые игры, не требующие мгновенной реакции. Например, подумайте, как реализовать игру в шахматы?
* Куда фантазия заведет (умные дома, управление автомобильной сигнализацией и т. д.

1. Что такое ****API?****

***API*** *(*от англ. ***A****pplication* ***P****rogramming* ***I****nterface*)

API - Описание способов взаимодействия одной компьютерной программы с другими.

Давайте представим себе, что мы с вами разработали какую-нибудь очень полезную программу. Программой пользуется все большее количество людей. У нас, конечно же, есть канал обратной связи, по которому пользователи сообщают нам об ошибках в программе и каких функций им не хватает. Мы исправляем ошибки и стараемся расширять функциональность. Но силы наши конечны, а пожеланий с ростом популярности программы становится все больше и больше. Делать все мы не успеваем. Да и нет смысла выполнять каждое пожелание, если мы понимаем, что данная функция нужна лишь небольшой аудитории. Мы можем написать библиотеку, дающую доступ к некоторым функциям нашей программы. Понятно, что таким образом мы дадим инструмент только программистам, но программиста можно нанять, и это уже не обязательно должен быть наш сотрудник. Если мы, помимо библиотеки, напишем инструкцию к ней, опишем ее функции, как ими пользоваться и как построить приложение на основе нашей библиотеки, то сторонний программист вполне сможет разобраться с ней и решить задачу заказчика.

И вот так мы как бы строим мостик между нашей программой и программистом, который хочет пользоваться ее возможностями. Мостик — это интерфейс. В англоязычной терминологии это называется Application Programming Interface или сокращенно API[эй-пи-ай]. API существует у большого количества программных продуктов: от операционной системы до интернет-сервисов. API — это обычный этап развития программных продуктов.

1. VCS (Version Control Systems)

***VCS (Version Control Systems) -*** (от англ. Система контроля версиями)

VCS **-** программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Представьте: мы работаем над новым хобби-проектом — программой, выводящей на экран изображение снежного человека. Как сделать процесс удобным и максимально продуктивным?

Когда появляются новые идеи, хочется быстрее их проверить, изменить код. Но важно при этом сохранить и промежуточные результаты, чтобы затем, если понадобится, к ним вернуться. Например, чтобы сравнить версии и выбрать более удачную.

В определенный момент становится ясно, какой должна быть финальная версия программы. Дальше работа продолжается вместе с другим программистом.

Когда каждый из нас захочет поэкспериментировать с результатом со своего устройства, у обоих должен быть доступ к данным.

Потом, когда оба программиста поработали отдельно, получившееся надо объединить в одно целое. И, конечно, совсем не хочется, чтобы все усилия пропали из-за сломавшейся техники.

Тогда на помощь приходят системы контроля версий. Что они делают:

1. Сохраняют код при поломках
2. Хранят много версий кода программы и позволяют легко переключаться между ними
3. Помогают разработчикам обмениваться кодом и редактировать один и тот же код с разных устройств
4. Объединяют результаты труда нескольких разработчиков

Чаще всего используют системы контроля версий Git, Mercurial (ее еще называют Hg) и несколько устаревшие Subversion (она же SVN) и CVS. У каждой из них есть свои плюсы и минусы, я же использовал систему **Git**.

1. Базы данных (SQLite).

Разработчики промышленного программного обеспечения столкнулись с проблемой хранения информации достаточно давно, и в качестве одного из решений еще в 1970 году [Эдгар Кодд](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%B4,_%D0%AD%D0%B4%D0%B3%D0%B0%D1%80) предложил реляционную модель данных (файловые базы данных появились еще раньше — в 1955 году). Эта идея развилась в привычные сегодня для почти каждого программиста реляционные базы данных (БД).

***Data base (База данных, БД)***

База данных — это непосредственное хранилище информации, которое без инструментов для взаимодействия с ним не очень-то и полезно. Такой интерфейс для общения с БД разработчикам и системным администраторам предоставляет специальное программное обеспечение — Системы управления базами данных (**СУБД**).

Существует достаточно много различных коммерческих и бесплатных СУБД. В своём проекте я буду использовать компактную встраиваемую реляционную СУБД SQLite по нескольким причинам:

1. SQLite — встраиваемая СУБД, поэтому не требует установки дополнительного программного обеспечения, а движок SQLite представляет собой отдельную библиотеку, написанную на C, которую можно использовать как составную часть вашей программы.
2. SQLite база данных представляет собой один файл, с которым удобно работать.
3. Исходный код SQLite передан в общественное достояние, то есть не существует никаких лицензионных ограничений на использование СУБД, как в некоммерческих, так и в коммерческих целях.
4. Большая часть дополнительных инструментов для работы с SQLite бесплатна.
5. И, наконец, в составе стандартной библиотеки Python уже содержится библиотека для работы с SQLite.

Но не надо думать, что раз «SQLite» имеет слово «lite» в названии, то это какая-то «игрушечная» СУБД, которая используется только для обучения и при создании «настоящего» программного обеспечения не используется. Простота и удобство встраивания SQLite привели к тому, что библиотека используется в браузерах, музыкальных плеерах и многих других программах, например: Skype, Viber, Яндекс Браузер, Google Chrome, Mozilla Firefox, Safari, Opera, Adobe Lightroom и т. д.

Данные в SQLite базе данных, как и в любых других реляционных БД, хранятся с помощью таблиц и связей между этими таблицами. Если говорить в терминах баз данных, то таблицы — это сущности, а связи — отношения между этими сущностями. Строго говоря, проектирование и использование баз данных — серьезная область знаний в информационных технологиях, существует несколько разных профессий, специалисты в каждой из которых отвечают за отдельные части жизненного цикла базы данных.

1. Python.

***Язык программирования***

Язык программирования — набор строгих правил, согласно которым компьютер может понимать команды и выполнять их. Текст программы, написанной на любом языке программирования, называется **программным кодом** (или просто кодом).

Языки программирования бывают двух основных типов: компилируемые и интерпретируемые. Если программа написана на компилируемом языке (например, Pascal или C++), перед ее выполнением нужно сначала полностью проверить на некоторые ошибки и перевести в более понятную для компьютера форму. Это делает специальная программа — компилятор.

Если программа написана на интерпретируемом языке (например, JavaScript или Python), она не переводится целиком в машинный код, а специальная программа — интерпретатор — идет по коду, анализирует и выполняет каждую отдельную команду. Такой подход придает языку особую гибкость и простоту в написании программ.

Язык Python, который я использую, разработал голландский программист Гвидо Ван Россум (Guido van Rossum) в 1991 году. Интересный факт, язык назван не в честь змеи-питона: Гвидо был большим фанатом британского комедийного сериала «Летающий цирк Монти Пайтона» (англ. Monty Python’s Flying Circus), и именно оттуда пришло название языка. В настоящее время в русском языке для обозначения используют два варианта — «Питон» и «Пайтон».

Python относится к интерпретируемым языкам программирования: чтобы запустить написанную на Python программу, нужен интерпретатор Python (его можно скачать с сайта [python.org](http://python.org/)). Я выбрал его из-за простоты написания кода и больших технических возможностей. Хотя несмотря на это, он обладает рядом минусов, например скорость вычислений.

1. Библиотеки.

***Библиотека***

Библиотека в программировании — сборник подпрограмм или объектов, используемых для разработки программного обеспечения, или ПО.

Это можно сравнить с разделением книг на главы, абзацы, слова — без такого деления трудно понять смысл. Этот принцип — первый важный этап в понимании сущности библиотеки.

Следующий этап заключается в возможности использовать часть кода одной программы из другой программы. Обычно это нужно, чтобы повторно использовать код: не писать одно и то же два раза. А еще — не переписывать код, написанный на другом языке программирования. Такое свойство часто называют модульностью.

**PyPI (pip)** — центральный репозиторий (хранилище) модулей для языка программирования Python. Он как PlayMarket для Android, AppStore для iPhone. Здесь можно скачать все нижеперечисленные библиотеки, что я использовал в написании своей программы.

* Requests
* PyTelegramBotAPI

# Глава 2. Разработка программного кода

1. main.py

Для написания программы я использовал стандартные библиотеки интерпретатора python (os, threading, time, datetime). А также дополнительные библиотеки, которые можно скачать с помощью **PyPI.**

Основной код программы содержится в файле main.py. Здесь происходить вывод кнопок навигации в функциях: start, interface, view\_questions. Регистрация пользователей в функции registration\_student. Создания варианта для проверки в функции create\_stasks и проверка отправленного учеником варианта в функции check\_question. Здесь же проверенные работы, зарегистрированные пользователи заносятся в базы данных (result\_base, telegram\_db соответственно).

1. to\_result\_db.py, to\_telegram\_db.py

Файлы нужные для внесения/считывания значений в/из базы данных.

1. config.py

Файл, в котором находятся пути, что используются в файле **main.py**.

1. B\_D

Папка для хранения вариантов, что задал учитель.

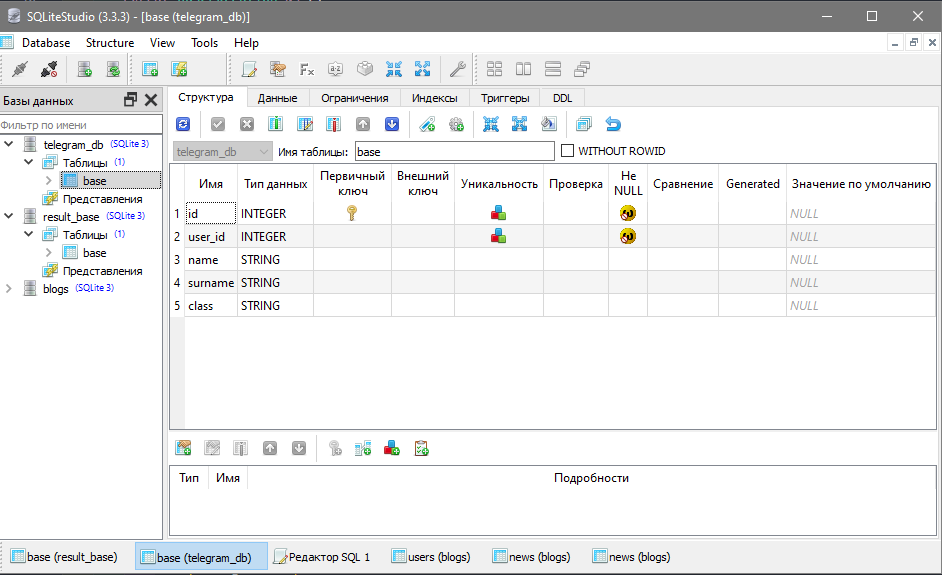
1. Base

Папка для хранения карточек (номеров задач) для создания вариантов.

1. Устройство базы данных

Таблицы баз данных имеют следующий формат:

**telegram\_db**



**id** – id данных

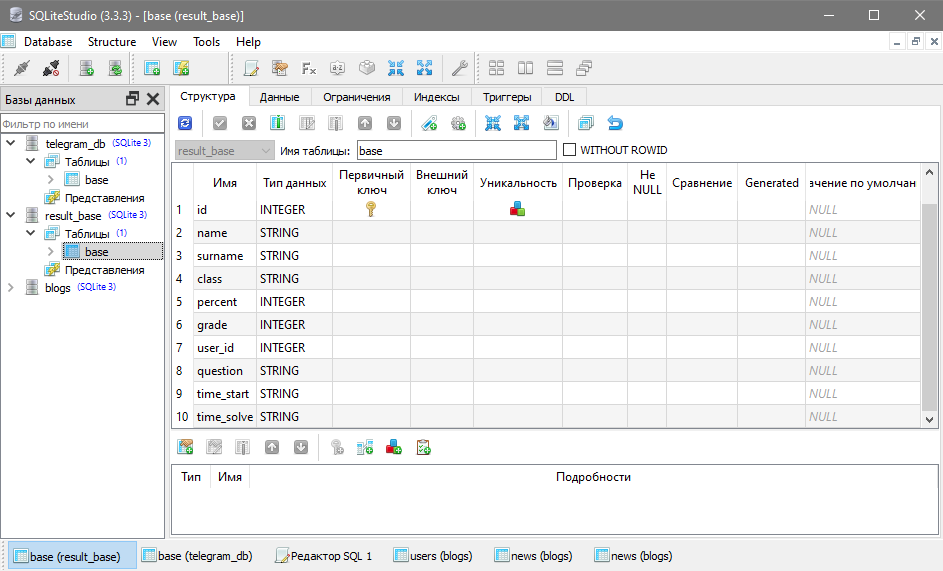
**user\_id –** id пользователя в мессенджере

**name –** Имя пользователя

**surname –** Фамилия пользователя

**class** – Класс обучения пользователя

**result\_base**



**id** - id данных

**name** - Имя пользователя

**surname** - Фамилия пользователя

**class** - Класс обучения пользователя

**percent** – Процент решения варианта

**grade** – Оценка, поставленная за вариант

**user\_id** - id пользователя в мессенджере

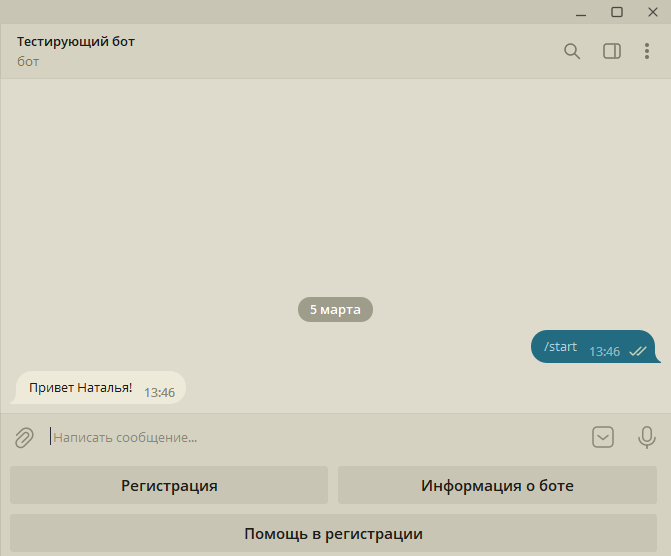
**question** – Вариант, который решал пользователь

**time\_start** – Время начала варианта

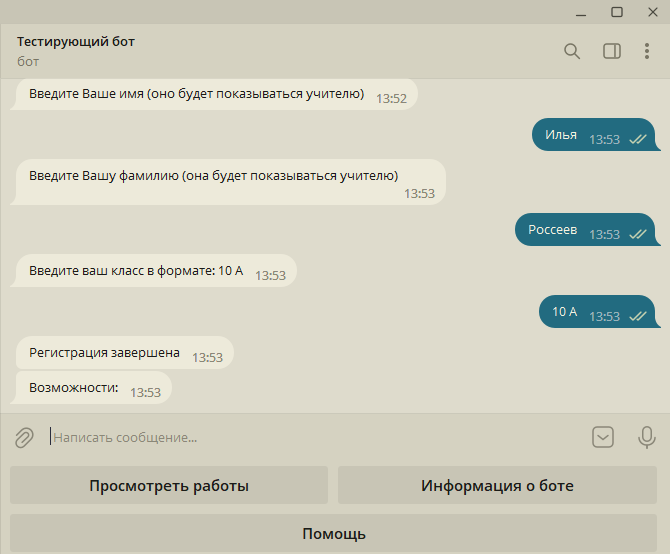
**time\_solve** – Время, которое пользователь потратил, чтобы решать вариант

# Глава 3. Устройство чат бота

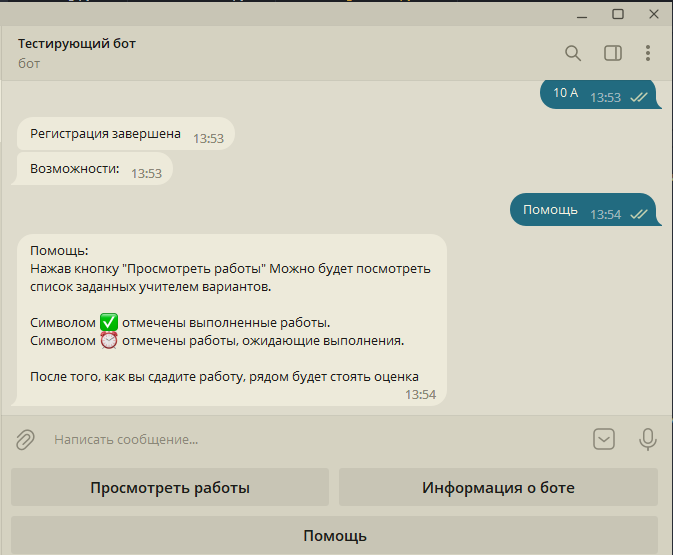
1. Окно регистрации



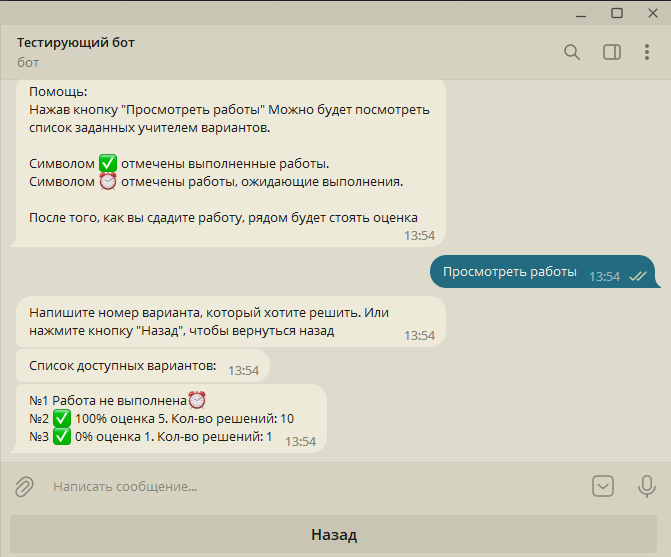
1. Основное окно



1. Окно помощи



1. Окно выбора варианта



1. Пример решения варианта

