Оглавление

[Оглавление 3](#_Toc42346038)

[Введение 4](#_Toc42346039)

[1. Аналитический раздел 5](#_Toc42346040)

[1.1 Постановка задачи 5](#_Toc42346041)

[1.2. Анализ систем управления базами данных 6](#_Toc42346042)

[1.3 Анализ алгоритмов работы чат-бота 10](#_Toc42346043)

[2. Конструкторский раздел 12](#_Toc42346044)

[2.1 Описание метода 12](#_Toc42346045)

[2.2 Конструирование базы данных 13](#_Toc42346046)

[2.3 Модель системы 14](#_Toc42346047)

[2.2 Структуры данных 17](#_Toc42346048)

[2.4 Машина состояний 18](#_Toc42346049)

[3. Технологический раздел 20](#_Toc42346050)

[3.1 Средства реализации 20](#_Toc42346051)

[3.2 Использование объектно-реляционного отображения 20](#_Toc42346052)

[3.3 Описание интерфейса чат-бота 24](#_Toc42346053)

[4. Исследовательский раздел 25](#_Toc42346054)

[Заключение 29](#_Toc42346055)

[Список использованных источников 30](#_Toc42346056)

Введение

Звонки по телефону поколение Z не любит больше всего. Чтобы ориентироваться на так называемых зумеров как на целевую аудиторию необходима выработка альтернативных способов коммуникации с целью поддержания связи с клиентами. Представим бар, проводящий большое количество мероприятий. Место популярное и на мероприятия приходит много человек, но для всех вместимости не хватает. Одно из возможных решений – регистрация на эти события. Для этого можно предложить много решений – звонки, сообщения в инстаграме, разработка сайта или чат-бот. Целью данного курсового проекта является разработка чат-бота на платформе мессенджера Telegram с подключением базы данных. Задачей чат-бота является осуществлять запись гостей на мероприятия. Все запросы пользователей должны обрабатываться с использованием базы данных. Чат-бот должен иметь следующие функции:

* + 1. Для клиентов –
       - Просмотр событий на текущей неделе;
       - Запись на событие;
       - Отмена записи на событие;
    2. Для администраторов –
       - Просмотр статистики по регистрациям.
       - Добавление новых событий
       - Редактирование существующих событий;

с) Для обеих ролей –

* Регистрацию нового пользователя;
* Возможность смены имени.

В ходе курсового проекта должны быть решены следующие задачи:

* + - Анализ существующих СУБД, выбор СУБД для этого проекта;
    - Проектирование программного обеспечения;
    - Проектирование архитектуры базы данных;
    - Тестирование программного обеспечения.

1. Аналитический раздел

1.1 Постановка задачи

Обязательные требования к программному обеспечению:

1) Необходимо разработать чат-бота на платформе мессенджера Telegram с подключением базы данных, то есть, необходимо реализовать две составляющие: бота(приложение) и базу данных.

2) Задачей чат-бота является предоставления сервиса клиентам и сотрудникам бара. То есть, программа должна исполнять свою основную задачу, а также необходима корректная работа программы.

3) Все запросы должны обрабатываться с использованием базы данных. То есть, должны быть разработаны запросы к базе данных.

4) Чат-бот должен иметь следующие функции:

* + 1. Для клиентов –
       - Просмотр событий на текущей неделе;
       - Запись на событие;
       - Отмена записи на событие;
    2. Для администраторов –
       - Просмотр статистики по регистрациям.
       - Добавление новых событий
       - Редактирование существующих событий;

с) Для обеих ролей –

* Регистрацию нового пользователя;
* Возможность смены имени.

5) Архитектура приложения и разработанный UI должны быть понятны и открыты к возможным будущим изменениям.

Разработка данного программного обеспечения делится на два основных этапа: проектирование базы данных и разработка приложения, посредством которого происходит обращение к этой базе данных и обработка запросов пользователей. Далее будут проанализированы и выбраны средства управления базами данных и алгоритмы работы приложения чат-бота. Также в этом разделе будет произведено построение модели информационной системы.

1.2. Анализ систем управления базами данных

Системы управления базами данных (далее СУБД) – приложение, обеспечивающее создание, хранение, обновление и поиск информации в базах данных.

По моделям данных СУБД классифицируются следующим образом:

a. Дореляционные

b. Реляционные

c. Постреляционные

Для данной задачи подойдет использование реляционной модели данных. Реляционная модель состоит из трех частей: структурной, целостностной и манипуляционной. Структурная часть реляционной модели описывает, из каких объектов состоит реляционная модель. В целостностной части реляционной модели фиксируются два базовых требования целостности, которые должны выполняться для любых отношений в любых реляционных базах данных. Это целостность сущностей и ссылочная целостность. Манипуляционная часть реляционной модели описывает два эквивалентных способа манипулирования реляционными данными – реляционную алгебру и реляционное исчисление. [1]

Таким образом, выбранная СУБД должна поддерживать реляционную модель данных. То есть, сущности должны храниться в таблицах (структурная часть), а также сущности разных таблиц должны иметь возможность быть связанными, то есть должна присутствовать возможность выстраивать отношения между сущностями при помощи внешних ключей (целостная часть). Также должна иметься возможность управлять данными, изменять их, производить поиск по данным и так далее (манипуляционная часть).

Рассмотрим три популярные на сегодняшний день СУБД, которые можно связать с данным проектом.

**CУБД SQLite.**

SQLite - это встраиваемая файловая СУБД, которая не требует установки или настройки. Это, в свою очередь, означает, что приложение не запускается в рамках отдельного серверного процесса, который необходимо запускать, останавливать или настраивать. Такая безсерверная архитектура позволяет базе данных поддерживать кроссплатформенность.

Полная база данных SQL содержится в одном файле на диске, и все операции чтения и записи осуществляются непосредственно в этот файл на диске. Поскольку данные напрямую записываются обратно в файл диска, SQLite поддерживает систему ACID защиты транзакций от сбоев выделения памяти и ошибок ввода-вывода диска, которые могут возникнуть в результате непредвиденных сбоев системы или сбоев питания. [2]

**Преимущества:**

1) Библиотека SQLite - одна из самых компактных библиотек СУБД, размер библиотеки может быть меньше 600 КБ.

2) Легко настраиваема и показывает хорошие результаты на веб-сайтах с низким и средним трафиком (~ 100 тыс. Запросов в день).

3) Совместима с языком программирования Python.

4) Файловая модель идеально подходит для выбранной задачи.

**Недостатки:**

1) Редко используется для создания многопользовательских приложений. При достаточно больших объемах записей могут возникнуть проблемы с согласованностью.

**СУБД MySQL**

MySQL - одна из самых популярных и широко распространенных СУБД с открытым исходным кодом. В отличие от SQLite, она использует архитектуру клиент-сервер, которая состоит из многопоточного сервера SQL. Эта многопоточная природа MySQL обеспечивает большую производительность, поскольку потоки ядра могут легко использовать несколько процессоров. База данных написана на C и C ++ и поддерживает различные платформы. Она также придерживается системы ACID для обеспечения согласованности транзакций. [2]

**Преимущества:**

1) Обширный функционал и наличие корпоративных функций.

2) Важным отличием MySQL от SQLite является поддержка многопользовательских функциями MySQL. Это и наличие корпоративных функций, делает эту СУБД идеально подходящей для распределенных приложений.

3) MySQL имеет преимущество перед PostgreSQL, когда речь идет о пропускной способности и производительности. Она также намного проще в установке и использовании и более широко используется в сравнению с PostgreSQL.

**Недостатки:**

1) Плохо проявляет себя на больших операциях SELECT и BULK INSERT.

2) Может быть излишней для выбранной задачи.

**СУБД PostgreSQL**

PostgreSQL - это СУБД с открытым исходным кодом, в которой особое внимание уделяется расширяемости и соответствию стандартам. Как и MySQL, PostgreSQL использует модель данных клиент-сервер, а процесс сервера, который обрабатывает связь с клиентом, управляет файлами и операциями базы данных, называется процессом postgres.

PostgreSQL обрабатывает параллельные клиентские сессии, «разветвляя» новый процесс для каждого соединения. Этот процесс отделен от основного процесса postgres, он создается и уничтожается во время жизненного цикла клиентского соединения. Написанный на C, Postgres также поддерживает ACID, функции и хранимые процедуры. В отличие от MySQL, PostgreSQL поддерживает материализованные представления (кэшированные представления), что ускоряет частый доступ к большим и активным таблицам. [2]

**Преимущества:**

1) Благодаря превосходной способности параллельной обработки PostgreSQL выходит на первое место (по сравнению с MySQL) при выполнении длинных SELECT.

2) PostgreSQL всегда считался лучшим для аналитических процессов, таких как хранилище данных.

3) Расширяемость базы данных PostgreSQL также делает ее идеальной для исследований и научных проектов.

**Недостатки:**

1) Механизм хранения PostgreSQL требует большой работы, он неоптимален в определенных случаях.

2) PostgreSQL очень энергоемкий в сравнении с другими СУБД. Поскольку PostgreSQL разветвляет процесс для новых клиентских соединений, это может занять до 10 МБ на соединение. Эта модель может занимать много памяти при одновременном подключении клиента по сравнению с моделью MySQL для потокового подключения.

3) Сложно устанавливать, много обновлений, не поддерживает кластеризованные индексы, что может оказать огромное негативное влияние на производительность по сравнению с базами данных MySQL.

**Обоснование выбора СУБД**

После анализа трех популярных СУБД был сделан выбор в сторону SQLite. Во-первых, эта СУБД поддерживает файловую модель данных, которую наиболее удобно использовать в данном проекте. Во-вторых, SQLite намного проще развернуть и она занимает меньше памяти, чем MySQL или PostgreSQL. В-третьих, она удовлетворяет всем требованиям безопасности и проектирования данного программного обеспечения. Также излишний функционал MySQL и PostgreSQL нерелевантен в данном проекте.

1.3 Анализ алгоритмов работы чат-бота

Чат-бот должен справляться со всеми задачами, описанными во введении посредством отправки вопросов, получения ответов на них и обработки полученной информации. Для обработки сообщений пользователей ботами на платформе мессенджера Telegram существует два вида алгоритмов: Webhooks — алгоритмы, работающие на основе механизмов оповещения системы о новых событиях и Long Polling — алгоритмы, работающие на основе опрашиваемых операций. Вне зависимости от используемого алгоритма все данные приходят в виде json-объектов, которые очень удобно обрабатывать.

Данные алгоритмы будут рассмотрены ниже, после чего будет выбран наиболее подходящий для решения текущей задачи.

**Long Polling — алгоритм опрашиваемых операций**

Это самый распространенный и просто реализуемый алгоритм, который используется при написании чат-ботов на платформе Telegram.

Суть этого алгоритма заключается в том, что бот периодически запрашивает обновления у Telegram, используя метод getUpdates [3], запрос должен отправляться не реже чем 1 раз в сутки, потому что дольше обновления не хранятся на серверах Telegram.

В этом алгоритме важно сбалансировать скорость ответа бота и ресурсы, затрачиваемые на опрос серверов Telegram.

Если слишком часто запрашивать обновления, скорость ответа будет высокой, но будет обрабатываться много бесполезных обновлений, не содержащих никакой новой информации. Если запрашивать обновления редко, будет очень большая задержка между отправкой сообщения пользователя и ответом бота.

**Webhooks — алгоритм оповещения системы о событиях**

В этом алгоритме нет необходимости для бота опрашивать серверы Telegram. Вместо этого сервера Telegram сообщают об обновлениях боту путем отправки в http-запросах структур данных, содержащих информацию о новых сообщениях. Webhook-и очень удобны, но также сложны в реализации. Для того, чтобы использовать этот алгоритм нужно иметь место на хостинге и зарегистрированный интернет домен. Кроме того хостинг должен поддерживать протокол https и SSL-сертификат.

**Обоснование выбора алгоритма чат-бота**

Для данного курсового проекта алгоритм long polling будет более подходящим, потому как будет происходить обработка исключительно текстовых данных, а также для его использования нет необходимости регистрировать интернет домен, с этим алгоритмом бота можно развернуть со своего компьютера. В данном проекте основной задачей остается реализация базы данных, поэтому можно пренебречь скоростью обработки сообщений.

2. Конструкторский раздел

Необходимо создать модель работы системы, которая будет связывать приложение и базу данных. Все операции ввода-вывода связаны с обращением к базе данных. Обработка и получение сообщений – задача библиотеки и алгоритмов Telegram бота.

Также необходимо создать машину состояний для того, чтобы сообщения обрабатывались корректно, таким образом алгоритмы приложения будут понимать, на каком этапе находится пользователь в данный момент, чтобы правильно считывать его сообщения и отвечать на них.

Далее будут рассмотрены описанные выше компоненты.

2.1 Описание метода

Когда начинает работать программа, приложение начинает собирать информацию об обновлениях с серверов Telegram с помощью метода getUpdates( ). Для каждого пользователя программа работает по-разному. Рассмотрим, как работает программа для одного конкретного пользователя.

Каждое сообщение обрабатывается после того, как бот получает очередное обновление. В обновлении содержится информация о том, какие новые сообщения пришли, о каждом сообщении известно, кем оно было отправлено, а также содержание этого сообщения.

Бот не начнет отвечать, пока не будет получена команда /start. Она обрабатывается следующим образом: если пользователь не зарегистрирован, то ему предлагается пройти регистрацию, после чего ему предлагается список других команд. Подробный алгоритм общения с пользователем будет приведен ниже.

После обработки сообщения, формируется ответное сообщение, которое отправляется пользователю при помощи метода sendmessage(id, message), где id – идентификатор пользователя в Telegram, а message – тектс сообщения.

2.2 Конструирование базы данных

База данных включает в себя три таблицы, которые изображены на рисунке 2.2.1.

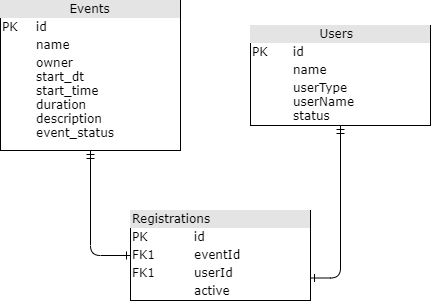


Рисунок 2.2.1 – Диаграмма таблиц базы данных

Таблица Users хранит информацию о пользователе и имеет следующие поля:

* id – идентификатор – primary key – в качестве него используется уникальный Telegram ID пользователя;
* name – имя гостя или администратора;
* userType – тип пользователя;
* userName – никнейм пользователя в Telegram (@nickname)
* status – статус пользователя (зарегистрирован, отключен и т.д.)

Таблица Events хранит информацию о событиях и имеет следующие поля:

* id – суррогатный primary key;
* name – название события;
* owner – никнейм владельца события;
* start\_dt – дата проведения;
* start\_time – время начала;
* duration – продолжительность;
* description – описание;
* event\_status – статус (состоится, отменен и т.д.)

Таблица Registration хранит информацию о записях, тем самым связывая таблицы пользователей и событий, и имеет следующие поля:

* id – суррогатный primary key;
* eventId – идентификатор события – внешний ключ;
* userId – идентификатор пользователя – внешний ключ;
* status – статус регистрации (активна или отменена).

ER-диаграмма базы данных показана на рисунке 2.2.2. Было выделено три сущности. Сущность Appointment связана с сущностями Patient и Doctor связью один ко многим. Атрибуты сущностей также отображены на диаграмме.

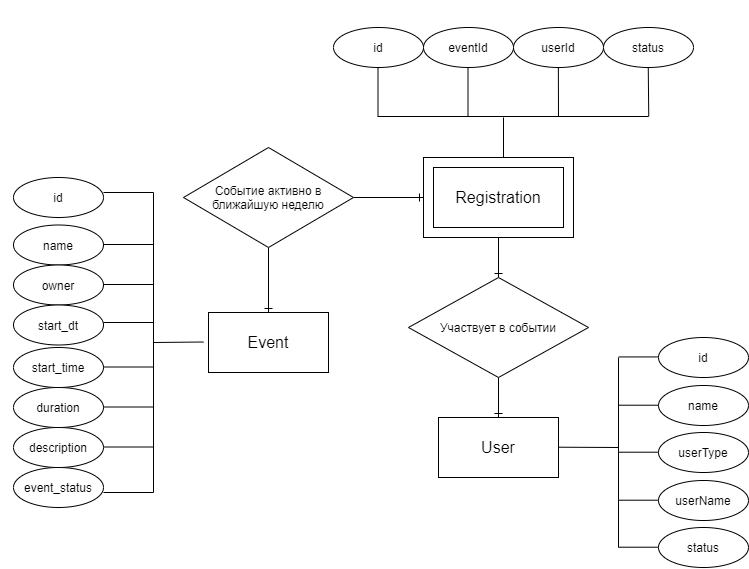


Рисунок 2.2.2 – ER-диаграмма модели

2.1 Диаграмма использования

Use-case диаграмма состоит из графической диаграммы, описывающей действующие лица и конкретные действия, которые выполняет пользователь при работе с системой. Данная диаграмма предназначена для определения функциональных требований. В системе есть три типа пользователей:

• незарегистрированный пользователь(гость): пишет команду /start – его уникальный Telegram ID сохраняется в базу данных со статусом PREGISTRADED, на этом этапе пользователю предлагается ввести имя, остальные функции не доступны;

• зарегистрированный пользователь, гость: использует приложение для просмотра предстоящих событий и управления регистрацией на них, имеют тип пользователя GUEST и статус REGISTRATED; переход в режим пользователя администратор осуществляется с помощью команды /admin и требует ввода пароля.

• зарегистрированный пользователь, гость: использует приложение для редактирования и создания новых событий, просмотра статистики. Имеет тип пользователя ADMIN и статус REGISTRATED;

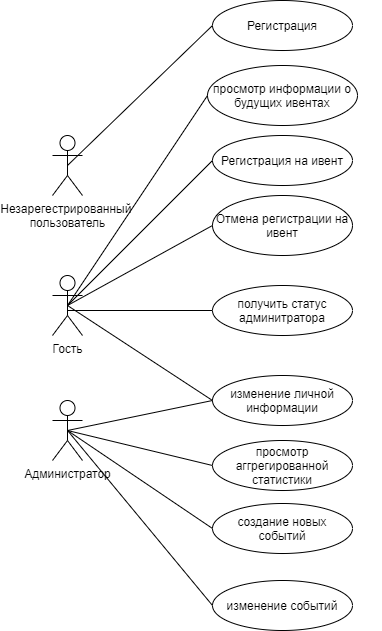


Рис 2.2.3 Use-case диаграмма

2.3 Модель системы

Модель на вход получает сообщения от пользователя, они обрабатываются при помощи использования алгоритмов чат-бота, полученные данные, передаются в базу данных, данные, которые в последствии отправляются пользователю, тоже берутся из базы данных, приложение формирует ответное сообщение, оно отправляется пользователю при помощи библиотеки Telebot.

Обращение к базе данных происходит каждый раз, когда требуется информация о врачах, пациентах или записях.

2.3 Структуры данных

В данном подразделе будут рассмотрены структуры данных, которые будут использованы в программе. В структуры входят:

1) Объектно-реляционное отображение базы данных классами:

* Класс пользователя User.
* Класс события Event.
* Класс записей Registration.

2) Message\_handlers – механизмы, позволяющие работать с сообщениями пользователя.

3) Callback\_query\_handlers – механизмы, позволяющие обрабатывать ввод пользователя с помощью inline buttons

4) Классы библиотеки Telebot, содержащие информацию о сообщениях пользователя.

5) Классы клавиатур Keyboard, позволяющие выводить на экран пользователя кнопки с доступными вариантами ответов.

6) Классы констант такие как UserStatus, UserType, EventStatus и т.д.

7) Классы GuestChoiceButton и AdminChoiceButton содержащие текстовые ответы на сообщения.

8) Статические параметры такие как dateFormat, timeFormat, API-token и т.д.

2.4 Модель данных

В данной работе будет использована технология ORM. ORM (англ. Object-Relational Mapping, рус. объектно-реляционное отображение) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». ORM позволяет удобно интегрировать модели в приложения с объектно-ориентированным стилем программирования.

3. Технологический раздел

**3.1 Средства реализации**

Для написания программного обеспечения для данного курсового проекта был использован язык Python 3.8. Этот язык поддерживает объектно-ориентированную модель разработки, что позволяет четко структурировать программу и легко модифицировать отдельные ее компоненты независимо от других.

В качестве среды разработки был использован PyCharm 2020.1 (build 201.6668.115). Он обладает всем необходимым функционалом для написания, профилирования и отладки программ.

Так как серверная программа не использует много ресурсов, достаточно одного потока для её работы.

3.2 Использование объектно-реляционного отображения

Для связывания серверной части и СУБД, было решено использовать объектно-реляционное отображение базы данных.

Объектно-реляционное отображение — это технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

Чтобы использовать этот метод работы с базой данных, было решено использовать классы и методы библиотеки Peewee.

Для работы с таблицами базы данных было создано три класса — каждый класс являлся представлением таблицы на ООП. Классы представлены в листингах 3.2.1 и 3.2.2.

class Events(Model):

id = AutoField(unique=True, null=False)

name = CharField()

owner = CharField()

start\_dt = DateField(formats=[dateFormat])

start\_time = TimeField(formats=[timeFormat])

duration = CharField()

description = CharField()

event\_status = IntegerField(default=None)

class Meta:

db\_table = 'events'

database = db

Листинг 3.2.1 – Класс события

class Users(Model):

id = PrimaryKeyField(unique=True, null=False)

name = CharField(default="")

userType = IntegerField(-1)

userName = CharField(default="")

status = IntegerField(default=UserStatus.UNKNOWN)

class Meta:

db\_table = 'users'

database = db

class Registrations(Model):

id = AutoField(unique=True, null=False)

eventId = ForeignKeyField(Events, backref='events')

userId = ForeignKeyField(Users, backref='users')

active = IntegerField()

class Meta:

db\_table = 'registrations'

database = db

Листинг 3.2.2 – Классы пользователя и регистрации

Данные классы соответствуют описанной в конструкторском разделе базе данных. Класс Registrstion связан с классами Users и Events посредством внешних ключей (ForeignKeyField). Поля классов соответствуют полям базы данных из конструкторского раздела. Были выбраны типы данных, соответствующие данным, хранимым в этих полях.

Для обращения к базе данных были реализованы функции записи и получения данных. Добавление данных представлено в листинге 3.2.3.

def add\_user(id, name="", userType=UserType.UNKNOWN, userName="", status=UserStatus.UNKNOWN):

Users.create(

id=id,

name=name,

userType=userType,

userName=userName,

status=status

)

return True

# true on success

def addRegistrationOnEvent(event\_id, user\_id) -> bool:

try:

Registrations.get(Registrations.eventId == event\_id,

Registrations.userId == user\_id,

Registrations.active== RegistrationStatus.ACTIVE)

except:

add\_registration(event\_id,

user\_id, RegistrationStatus.ACTIVE)

return True

def add\_registration(eventId=0,

userId=0, active=RegistrationStatus.INACTIVE):

app = Registrations.create(

eventId=eventId,

userId=userId,

active=active

)

app.save()

return True

def add\_event(name="", owner="", start\_dt="", start\_time="",

duration="",

description="", event\_status=EventStatus.UNKNOWN):

Events.create(

name=name,

owner=owner,

start\_dt=start\_dt,

start\_time=start\_time,

duration=duration,

description=description,

event\_status=event\_status

)

return True

Листинг 3.2.3 – Добавление записей

В листинге 3.2.4 представлены функции получения таблиц и удаления записей при помощи SQL запросов.

def get\_table(Class):

app\_table = (Class.select())

return app\_table

def get\_patients\_appointments(patid):

new\_table = Appointment.select().where(Appointment.patid == patid).execute()

return new\_table

def get\_clients\_by\_date(year, month, day, docid):

new\_table = Appointment.select().where(Appointment.docid == docid and Appointment.year == year and Appointment.month == month and Appointment.day == day)

return new\_table

def delete\_appointment(id):

Appointment.delete().where(Appointment.id == id).execute()

return 0

def delete\_patient(id):

Patient.delete().where(Patient.id == id).execute()

return 0

def delete\_doctor(id):

Doctor.delete().where(Doctor.id == id).execute()

return 0

Листинг 3.2.4 – Добавление записей

3.3 Описание интерфейса чат-бота

При первом обращении к чат-боту появляется сообщение о том, что необходимо пройти регистрацию перед использованием функций бота. Предлагается выбрать тип аккаунта: врач или пациент. Потом появится меню команд пациента. Если выбрать “Записаться к врачу” бот предложит список доступных специалистов. После этого нужно выбрать дату и время. После этого появится сообщение о подтверждении записи и поле выбора команд. Если выбрать “Отмена записи”, то выведется клавиатура с актуальными записями, где можно выбрать, какую запись отменить. Просмотреть все актуальные записи можно, выбрав пункт меню “Посмотреть актуальные записи.

Если в момент регистрации выбрать аккаунт врача, то начнется запросится информация о его специальности и ФИО, затем появится меню доступных действий. На данный момент единственным доступным действием для врача является просмотр записей по дате. Если записи в выбранную дату есть, то они выведутся на экран, если нет – выведется сообщение об этом.

4. Исследовательский раздел

Сервер-процесс занимает примерно 20 Мб оперативной памяти системы и почти не нагружает процессор.

Результаты работы чат-бот представлены на рисунках 4.1 – 4.8.

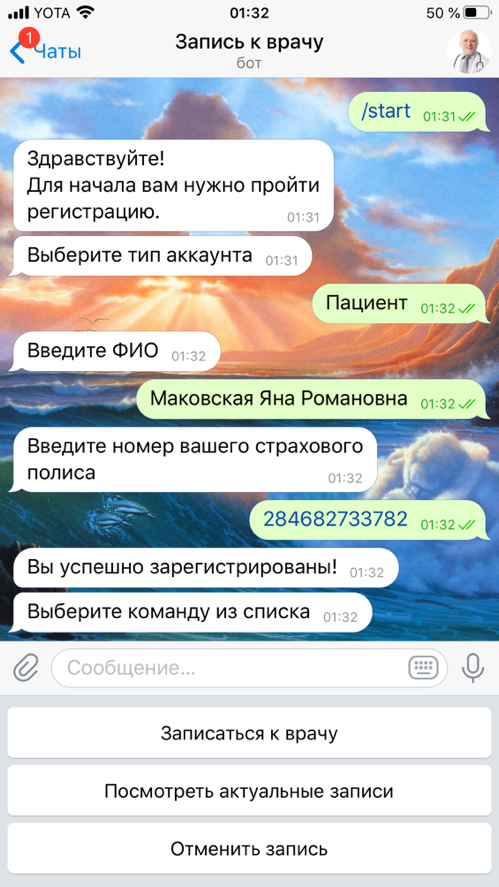
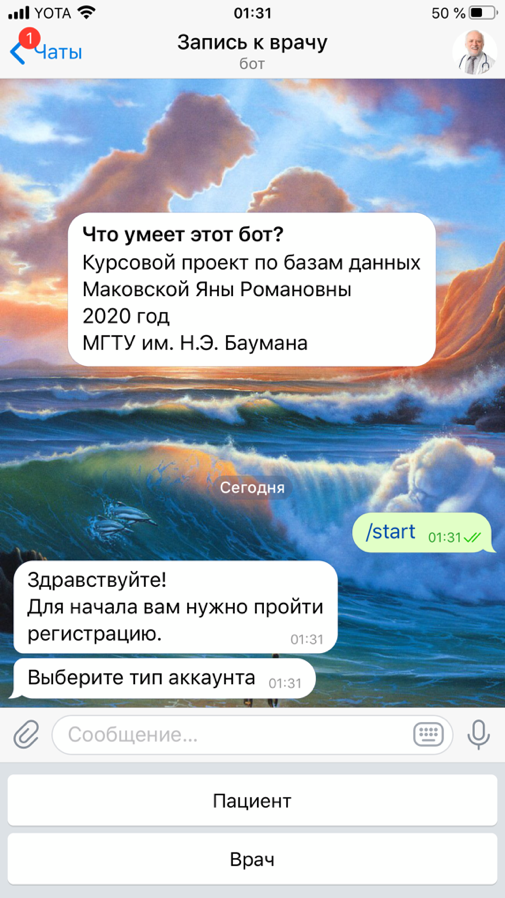


Рисунок 4.1 – Скриншоты регистрации пользователя

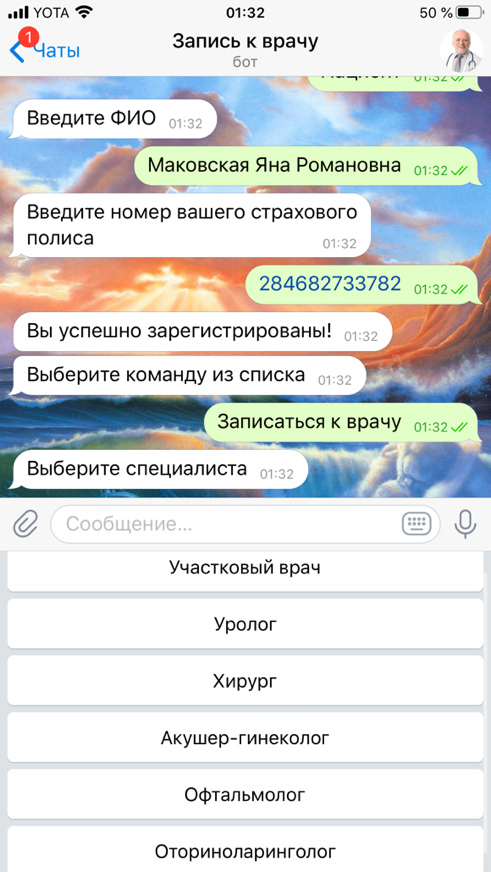


Рисунок 4.2 – Выбор специалиста

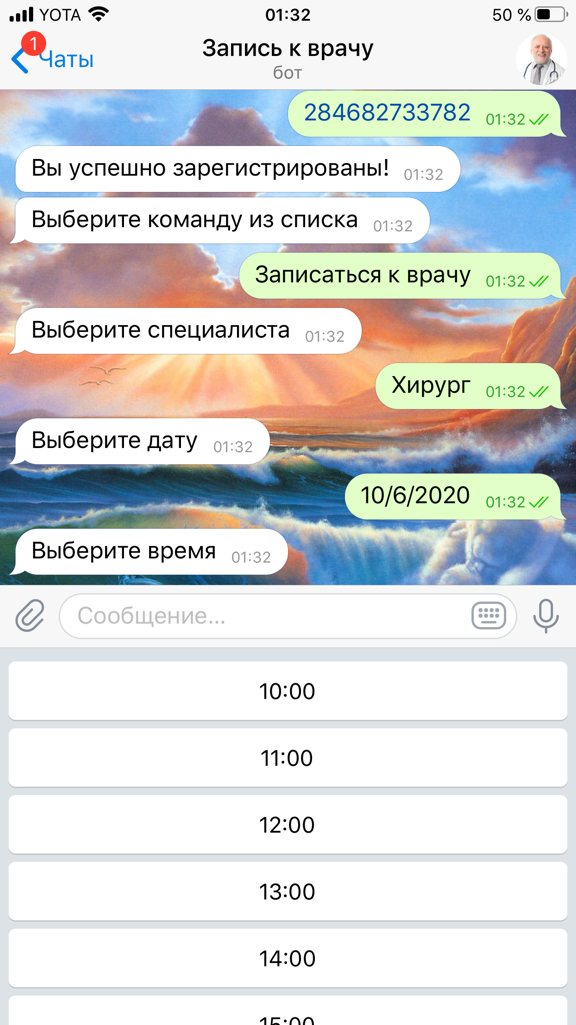
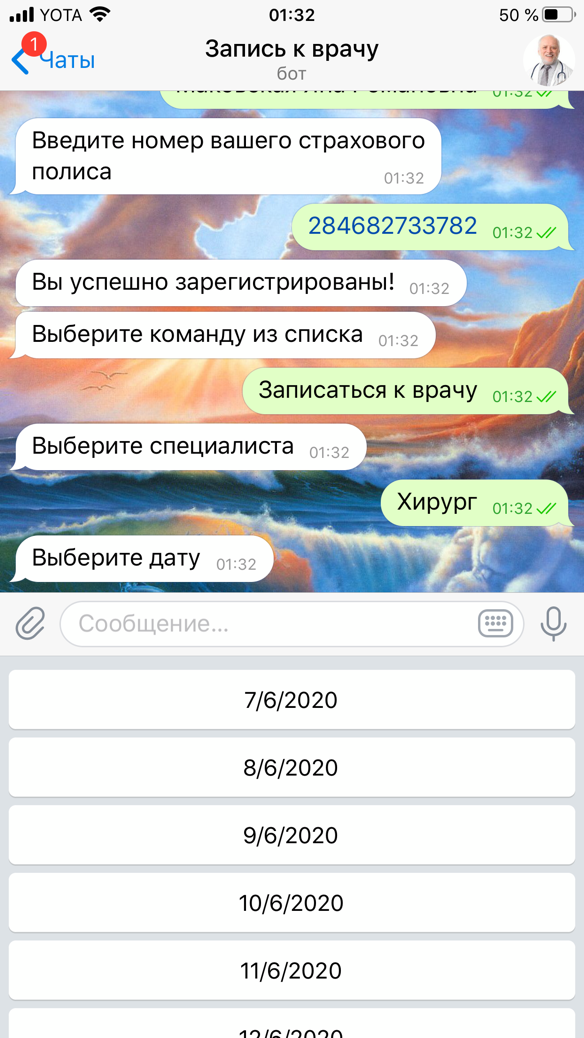


Рисунок 4.3 – Выбор даты и времени приема

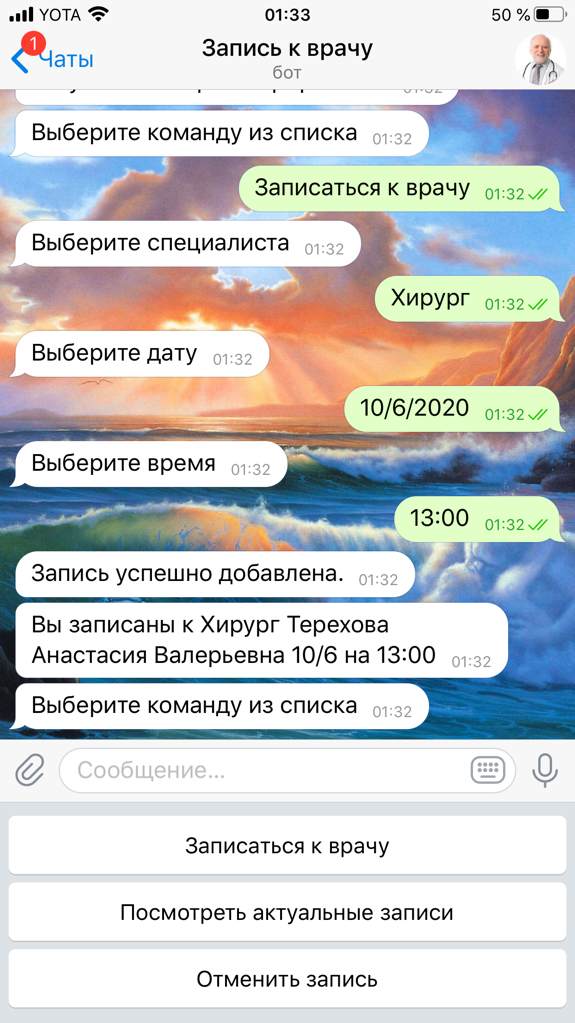


Рисунок 4.4 – Подтверждение записи, вывод команд

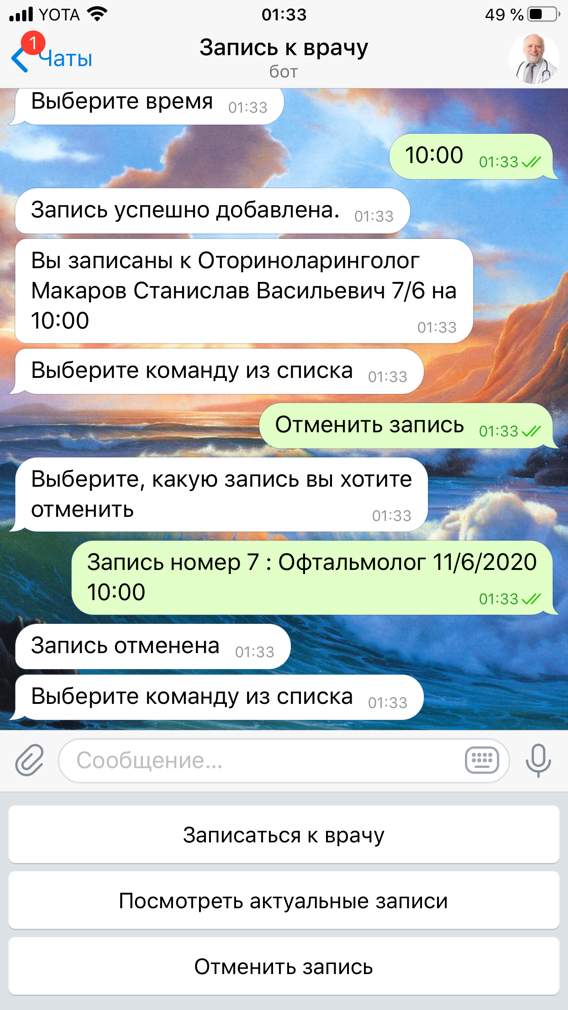
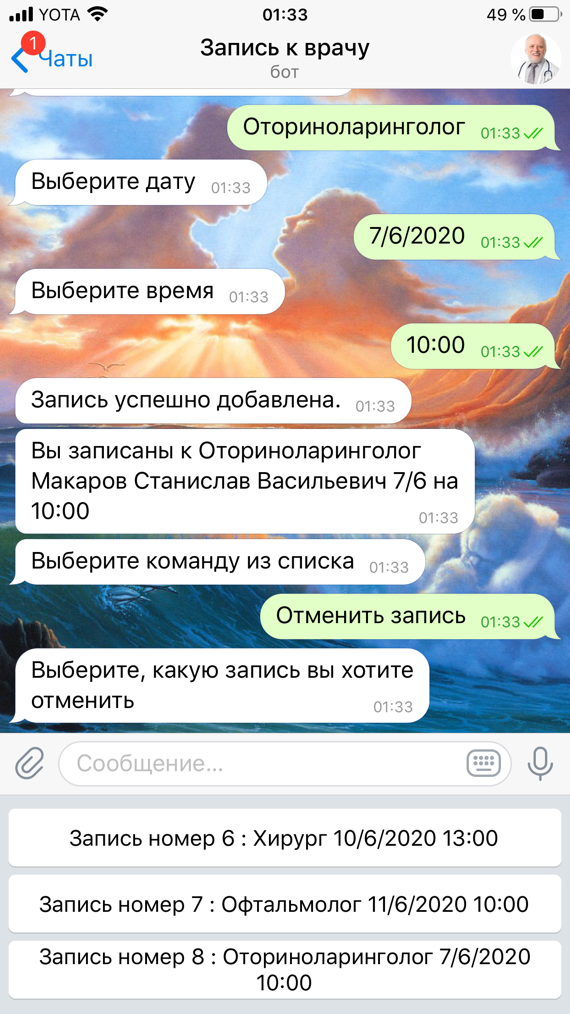


Рисунок 4.5 – Отмена записи

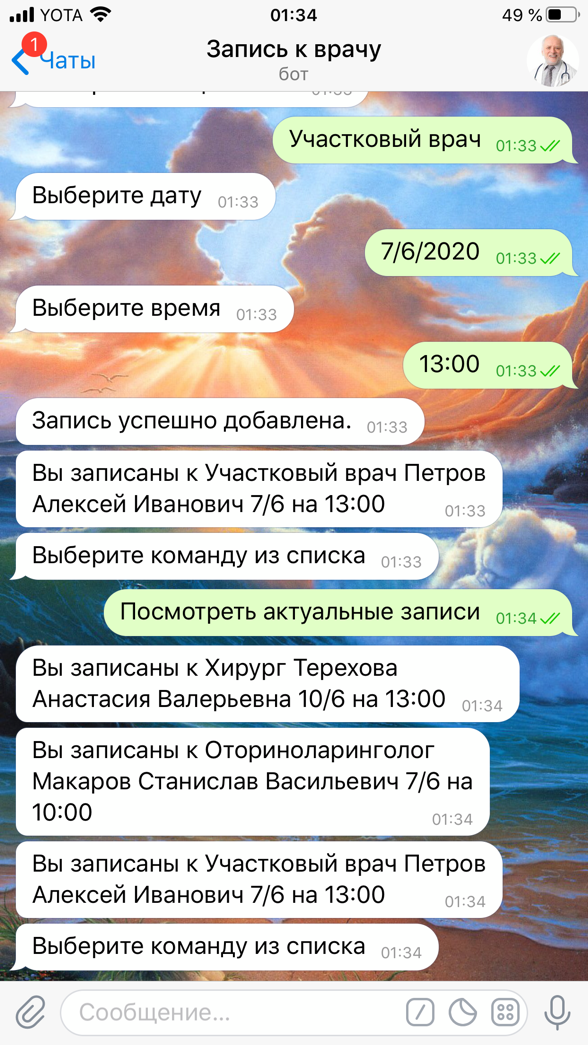


Рисунок 4.6 – Просмотр всех актуальных записей пациента

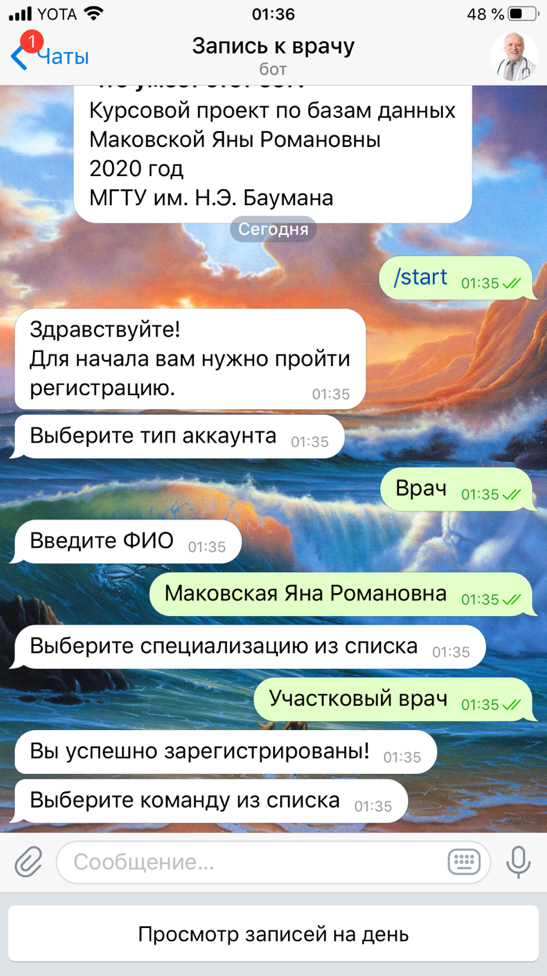


Рисунок 4.7 – регистрация и меню врача

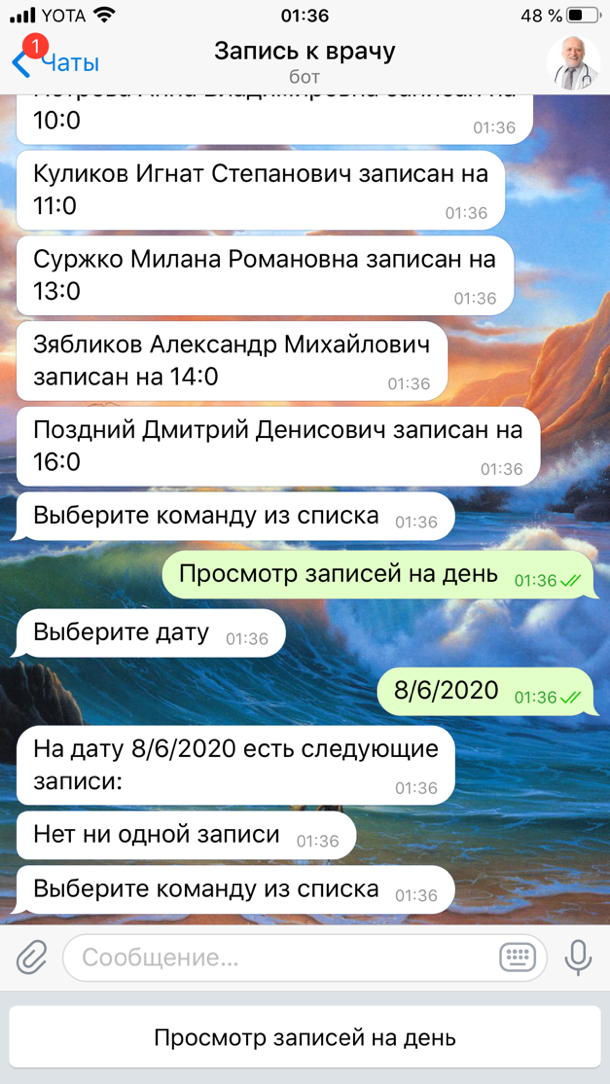
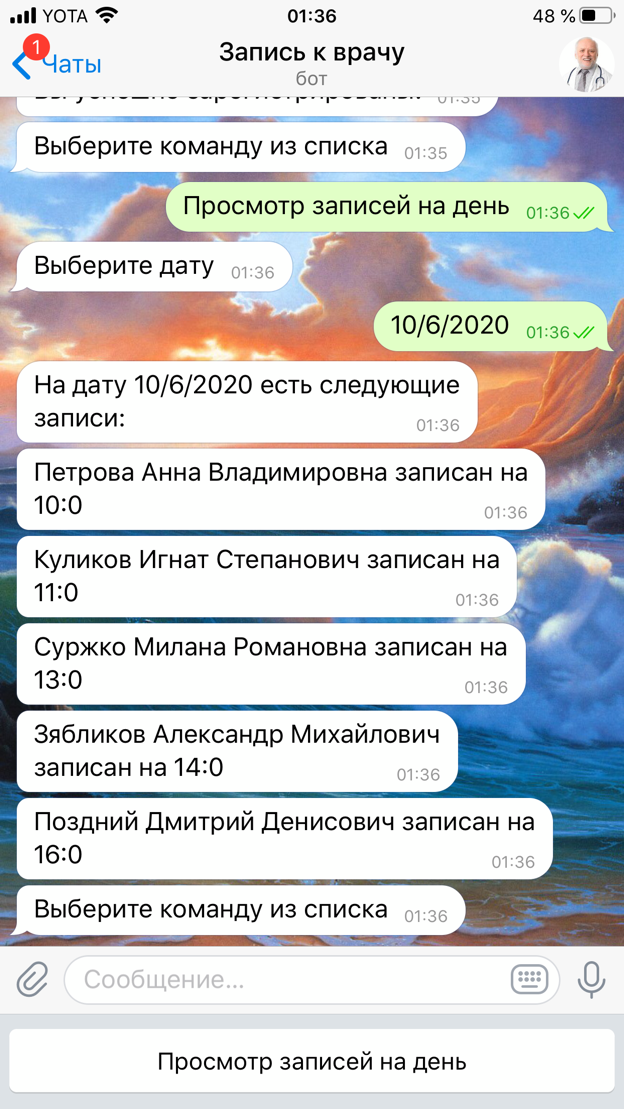


Рисунок 4.8 – Просмотр записей на день

Заключение

В результате выполнения курсового проекта была выполнена цель по разработке чат-бота на платформе мессенджера Telegram с подключением базы данных, задачей которого является обслуживание клиентов и сотрудников медицинских учреждений. Все запросы пользователей обрабатываются с использованием базы данных.

Были выполнены задачи поставленные во введении. Был выполнен анализ СУБД и анализ алгоритмов работы чат-ботов на платформе Telegram. Было спроектировано программное обеспечение, созданы схемы и диаграммы. Была спроектирована архитектура базы данных. Выбранные алгоритмы и структуры были реализованы в полной мере. Были произведены тестирование и отладка программного обеспечения.

В ходе данного курсового проекта были получены навыки создания чат-ботов на платформе Telegram, использования библиотеки Python Peewee для взаимодействия с СУБД SQLite.

Список использованных источников

[1] Гаврилова Юлия Михайловна, лекции из курса “Базы данных, ИУ7, 3 курс” — 2019 г.

[2] Интернет-ресурс Logz.io – сравнение характеристик популярных СУБД [Электронный ресурс]. – Режим доступа: URL: https://logz.io/blog/relational-database-comparison/ – (Дата обращения: 01.05.2020).

[3] Документация библиотеки работы с Telegram API — Telebot [Электронный ресурс]. – Режим доступа: URL: <https://core.telegram.org/bots/api> – (Дата обращения: 01.05.2020).

[4] Документация Python 3.8.3 Режим доступа: URL: <https://docs.python.org/3/>– (Дата обращения: 01.05.2020).

[5] Документация PyCharm [Электронный ресурс]. Режим доступа: URL: <https://www.jetbrains.com/ru-ru/pycharm/documentation/> – (Дата обращения: 01.05.2020).

[6] ORM Peewee [Электронный ресурс]. – Режим доступа: URL: <http://docs.peewee-orm.com/en/latest/index.html> – (Дата обращения: 16.05.2020).

[7] Документация СУБД SQLite [Электронный ресурс]. – Режим доступа: URL: https://www.sqlite.org/index.html – (Дата обращения: 06.05.2020).