ТЕЛЕГРАМ-БОТ ДЛЯ ОРГАНИЗАЦИИ ПРЕДЗАКАЗОВ В СЕТИ КОФЕЕН

Д.П. Пушкарева, *студент*
Научный руководитель: *Л.В. Макуха, старший преподаватель*
Сибирский федеральный университет
(Россия, г. Красноярск)

DOI:10.24412/2500-1000-2023-6-3-139-146

Аннотация. *В статье приводится описание работы телеграм-бота для оформления заказов в сети кофеен различных позиций меню с помощью мессенджера Telegram. Бот включает в себя выбор филиала, время посещения, ознакомление с меню и формирование предзаказа. Телеграм-бот значительно упрощает процесс оформления заказов в кофейне, позволяя пользователям совершать заказы и получать информацию о статусе заказа прямо через мессенджер Telegram. Это удобно для клиентов и помогает кофейне эффективно обрабатывать заказы.*

Ключевые слова: *мессенджер, телеграм-бот, бот, Telegram, кофе, кофейня, заказ, предзаказ.*

Самый важный ресурс сейчас – это время. Каждая минута обычного человеческого дня имеет свой вес и кажется, что всякий раз, когда хочется перекусить, слишком много времени теряется в очереди, особенно в популярных заведениях или в часы пик. И чтобы не тратить такой драгоценный ресурс можно и стоит использовать свое время рационально, пользуясь современными нововведениями в сфере общепита.

У большинства людей сейчас имеется современный смартфон, и почти у каждого есть различные мессенджеры, такие как: WhatsApp, VK, Viber, в том числе Telegram [1]. Например, мессенджер Telegram имеет большой спектр возможностей: общение с другими пользователями, развлекательный контент, рабочее пространство, с помощью которого можно организовать работу бота-предзаказов.

Преимущества использования телеграм-бота для предварительного заказа еды очевидны. Во-первых, можно оформить заказ в любое удобное время и из любого места, где есть подключение к Интернету. Теперь больше не нужно тратить время на поездку или ожидание в очереди, просто следует выбрать блюда, указать предпочтения и отправить заказ через бота.

Во-вторых, телеграм-боты обеспечивают удобство и скорость обработки заказов. Выбирая блюда заранее, пользователь избегает ненужного ожидания заказа, заказ будет готов к его приезду, и он сможет забрать его без лишних задержек.

Кроме того, телеграм-боты могут предлагать дополнительные функции, такие как уведомления о специальных предложениях или рекламных акциях, меню и часах работы заведения. Это поможет конечному потребителю быть в курсе последних обновлений и выбрать оптимальное время для посещения заведения.

Таким образом, использование телеграм-ботов для предварительного заказа еды – эффективный способ управлять своим временем и экономить его ресурсы [2].

Для того, чтобы оформить предзаказ, понадобится всего пара минут и смартфон, который всегда находится под рукой у пользователя, вместо более длительного ожидания очереди и приготовления своего заказа [3].

В процессе реализации телеграм-бота были поставлены и решены такие задачи, как проведение анализа предметной области и существующих аналогов, осуществление выбор средств разработки для сервиса, выполнение моделирование и программной реализации разрабатываемого телеграм-бота.

Для понимания работы телеграм-бота следует ознакомиться с диаграммой прецедентов, представленной на рисунке 1.

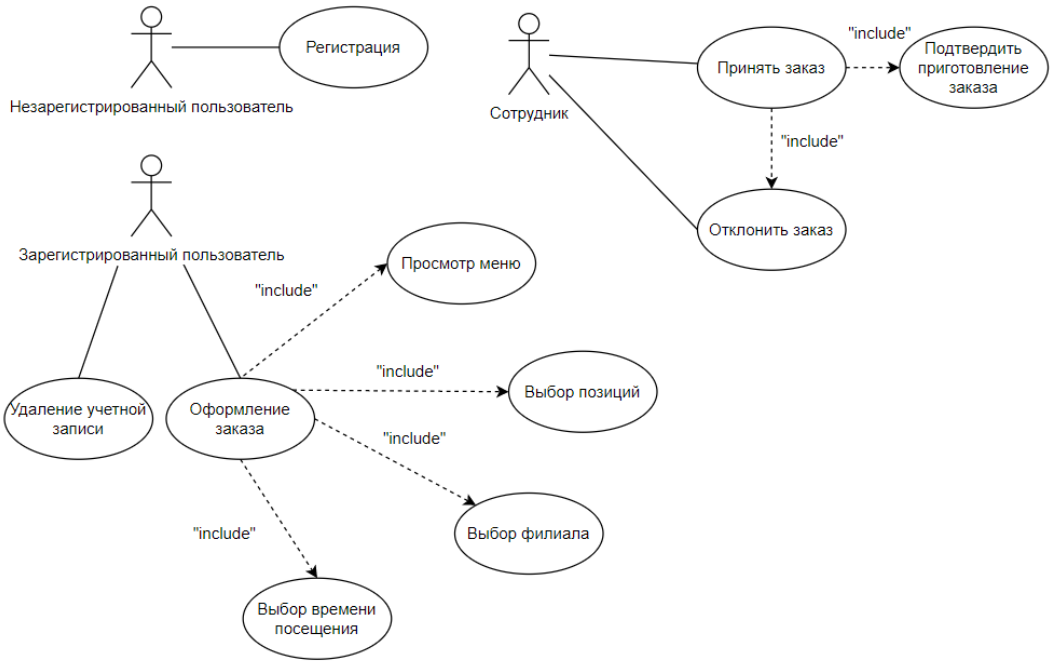

Рис. 1. Диаграмма прецедентов

В системе предусмотрено три роли пользователя:

- Незарегистрированный пользователь может пройти только процедуру регистрации, после чего он становится зарегистрированным пользователем, а его данные заносятся в соответствующую таблицу пользователь базы данных;

- Зарегистрированный пользователь имеет функцию оформить заказ, которая включает в себя выбор филиала, просмотр меню, выбор позиций и выбор времени посещения. Полученные данные записываются в таблицу заказов базы данных. Также пользователь может удалить свою учетную запись;

- Сотрудник уже работает над статусом предзаказа, который отправил пользователь. Он может принять предзаказ, подтвердить его приготовление и отклонить заказ.

Стоит ознакомиться подробнее с базой данных, которая представлена на рисунке 2. Данная БД содержит в себе две таблицы**: заказы** и **пользователи**.

Таблица пользователей содержит в себе информацию и о зарегистрированных пользователях, такие как: ID пользователя, его имя, номер телефона и имя пользователя в мессенджере Telegram. ID пользователя является его уникальным идентификатором. Имя пользователя используется для удобства обращения к гостю. Номер телефона и имя пользователя в Telegram нужны для связи с гостем для того, чтобы уточнить некоторые моменты, которые не были указаны в предзаказе.

Таблица заказов содержит больше информации. ID заказа является ID пользователя. Так как предусмотрено, что данный бот может использоваться в сети кофеен, то в БД также записывается филиал, который гость выбрал при оформлении заказа. Данные предзаказа, которые пользователь отправляет через бота, также хранятся в соответствующей записи, как и выбор времени посещения кофейни. В БД предусмотрена запись о статусе заказа и времени его совершения, для обратного отсчета выполнения заказа.

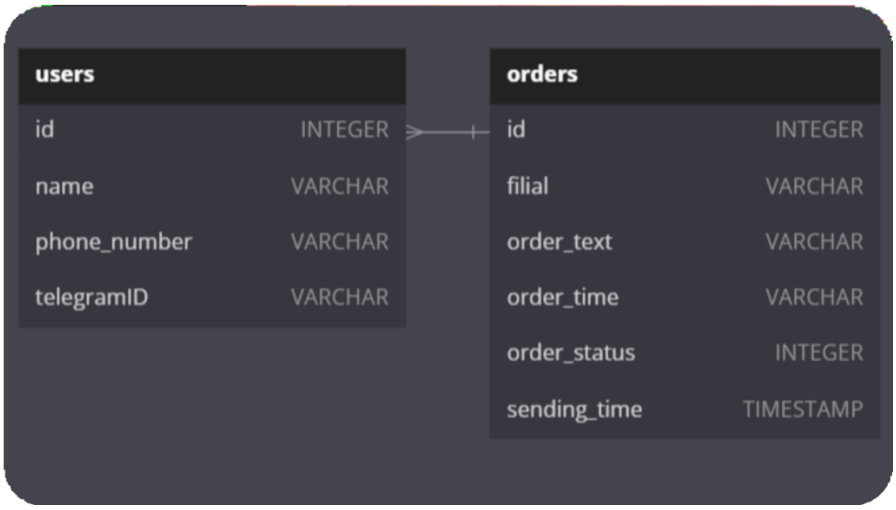

Рис. 2. База данных

Процесс создания, заполнения и поиска данных полностью осуществлен с помощью кода, который представлен на рисунках 3-4.

```
data = cursor.execute("select count(*) from sqlite_master where type='table' and name='users'")
for row in data:
    if row[0] == 0:
        cursor.execute('CREATE TABLE users(id INTEGER, name VARCHAR, phone_number VARCHAR, telegramID VARCHAR)')
        database.commit()

def add_user(message):
    cursor.execute("SELECT id FROM users WHERE id=?",(message.chat.id,))
    user = cursor.fetchone()
    if not user:
        cursor.execute("INSERT INTO users VALUES(?,?,?,?)",(message.chat.id,"name","phone_number","telegramID",))
        database.commit()
    else:
        return False

def drop_user_reg(user_id):
    cursor.execute("DELETE FROM users WHERE id=?",(user_id,))
    database.commit()

def add_user_name(message):
    cursor.execute("UPDATE users SET name=? WHERE id=?",(message.text,message.chat.id))
    database.commit()

def add_user_telegramID(message):
    cursor.execute("UPDATE users SET telegramID=? WHERE id=?",(message.text,message.chat.id))
    database.commit()

def add_user_numb(message):
    cursor.execute("UPDATE users SET phone_number=? WHERE id=?",(message.text,message.chat.id))
    database.commit()

def get_user_name(user_id):
    cursor.execute("SELECT name FROM users WHERE id=?",(user_id,))
    user_name = cursor.fetchone()[0]
    return user_name
def get_user_numb(user_id):
    cursor.execute("SELECT phone_number FROM users WHERE id=?",(user_id,))
    user_numb = cursor.fetchone()[0]
    return user_numb
def get_user_tgID(user_id):
    cursor.execute("SELECT telegramID FROM users WHERE id=?",(user_id,))
    user_tgID = cursor.fetchone()[0]
    return user_tgID
```

Рис. 3. Таблица пользователей в БД

```
data = cursor.execute("select count(*) from sqlite_master where type='table' and name='orders'")
for row in data:
    if row[0] == 0:
        cursor.execute('CREATE TABLE orders(id INTEGER, filial VARCHAR, order_text VARCHAR, order_time VARCHAR, order_status INTEGER, sending_time timestamp)')
        database.commit()

def create_order(message):
    cursor.execute("SELECT id FROM orders WHERE id=?",(message.chat.id,))
    order = cursor.fetchone()
    if not order:
        cursor.execute("INSERT INTO orders VALUES(?,?,?,?,?,?)",(message.chat.id,"filial","order_text","order_time","order_status","sending_time"))
        database.commit()
    else:
        return False

def choise_filial(message,filial):
    cursor.execute("UPDATE orders SET filial=? WHERE id=?",(filial,message.chat.id))
    database.commit()

def add_order(message, order):
    cursor.execute("UPDATE orders SET order_text=? WHERE id=?",(order,message.chat.id))
    database.commit()

def add_time(message, time):
    cursor.execute("UPDATE orders SET order_time=? WHERE id=?",(time,message.chat.id))
    database.commit()

def status_order(message,status):
    cursor.execute("UPDATE orders SET order_status=? WHERE id=?",(status,message.chat.id))
    database.commit()

def sending_time(message, sending_time):
    cursor.execute("UPDATE orders SET sending_time=? WHERE id=?",(sending_time,message.chat.id))
    database.commit()

def get_filial(user_id):
    cursor.execute("SELECT filial FROM orders WHERE id=?",(user_id,))
    order_filial = cursor.fetchone()[0]
    return order_filial

def get_order_text(user_id):
    cursor.execute("SELECT order_text FROM orders WHERE id=?",(user_id,))
    order_text = cursor.fetchone()[0]
    return order_text

def get_time(user_id):
    cursor.execute("SELECT order_time FROM orders WHERE id=?",(user_id,))
    order_time = cursor.fetchone()[0]
    return order_time
```

Рис. 4. Таблица заказов в БД

При регистрации пользователя создается запись в таблице, если она еще не существуют, либо «привязывается» к уже существующей. ID пользователя является уникальным значением и не меняется даже при повторной регистрации, именно поэтому все заказы хранят это значение для истории обращений в БД.

Процедура регистрации достаточно проста. Пользователь прописывает команду /start, после чего бот требует ввести некоторые данные. Если же у пользователя уже есть учетная запись, то бот выводит его данные для проверки, а после предлагает оформить предзаказ, используя команду /order, либо же удалить свою учетную запись с помощью команды /restart. Данный функционал бота представлен на рисунке 5.

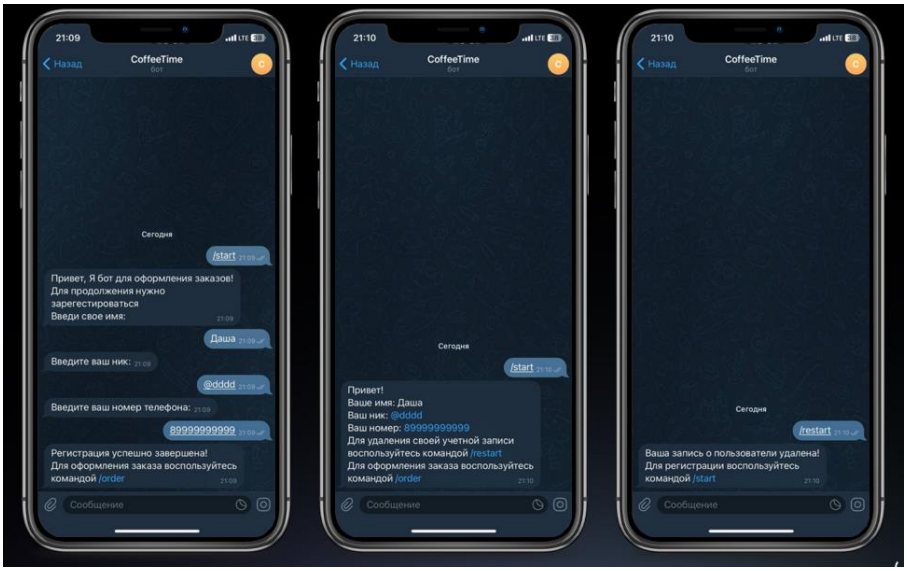

Рис. 5. Учетная запись пользователя

Для оформления заказа необходимо прописать команду /order и выбрать необходимый филиал, который предложит бот. Сразу после появится навигационное меню, в котором представлено меню выбранного филиала, кнопка возврата и кнопка оформления заказа. Так как в боте предусмотрено, что позиции в меню каждого из филиалов могут отличаться, соответственно, за каждым филиалом закреплено свое индивидуально меню, что продемонстрировано на рисунке 6.

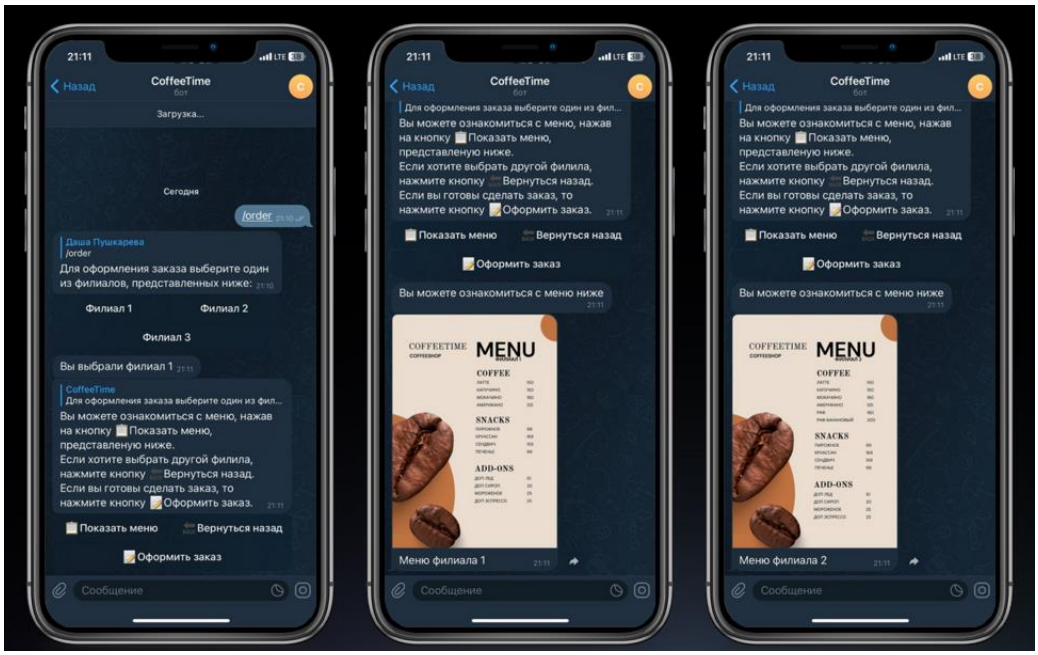

Рис. 6. Оформление заказа, навигационное меню

Следующим шагом является этап оформления заказа. После того, как пользователь ознакомился с меню и выбрал необходимые нужные позиции, происходит информирование бота о конечном выборе пользователя. При нажатии на кнопку «Оформить заказ», бот отправляет сообщение с правилами оформления заказа, а именно, что одна позиция вводится одним текстовым сообщением, для начала ввода позиций следует нажать на кнопку «Добавить позицию», а лишь после отправлять сообщение. Для добавления еще одной позиции нужно нажать соответствующую кнопку. После ввода всех позиций, осуществляется переход к следующему этапу при нажатии кнопки «Продолжить». Стоит отметить, что в боте предусмотрены ограничения: при нажатии на кнопку «Продолжить» без ввода хотя бы одной позиции, бот сообщит об этом и вернет пользователя обратно к навигационному меню, которое упоминалось ранее. Также, если пользователем вводится более чем 20 позиций, то бот выведет соответствующее сообщение – Превышен лимит позиций. Работа бота и демонстрация функций представлены на рисунках 7-8.

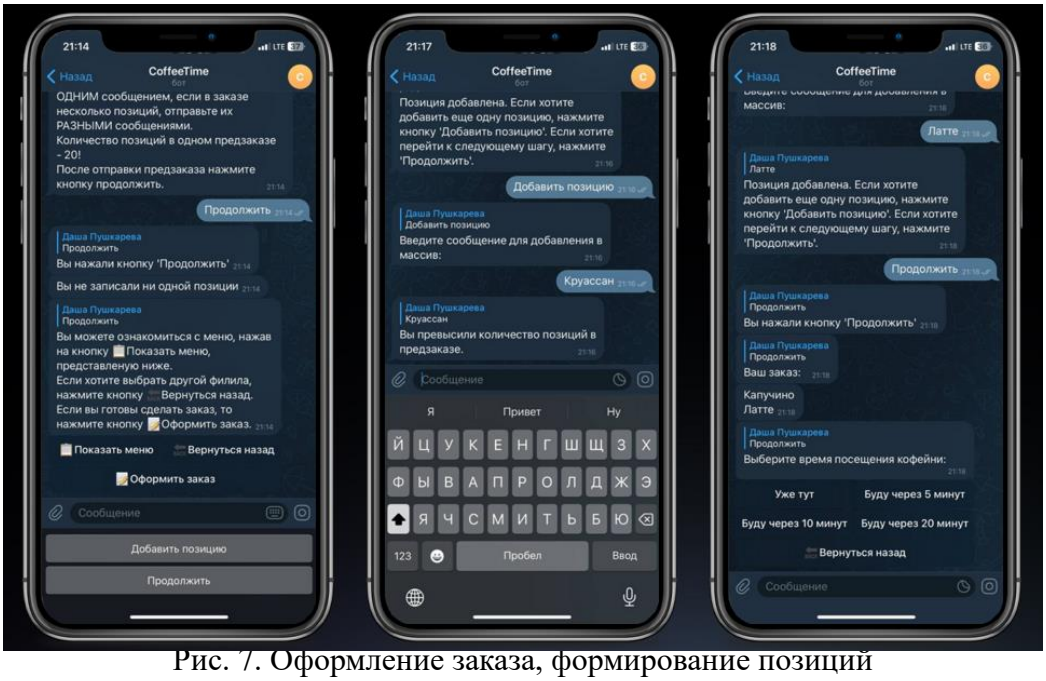

Рис. 7. Оформление заказа, формирование позиций

```python
@dp.message_handler(state=SomeState.ADD_POSITION, content_types=types.ContentTypes.TEXT)
async def process_position(message: types.Message, state: FSMContext):
    global messages
    message_text = message.text

    if len(messages) >= 20:
        await message.reply("Вы превысили количество позиций в предзаказе.")
    else:
        messages.append(message_text)

    await message.reply("Позиция добавлена. Если хотите добавить еще одну позицию, нажмите кнопку 'Добавить позицию'.
    await state.finish()

async def history(message: types.Message, state: FSMContext):
    chat_id = message.chat.id

    if messages:
        await message.reply("Ваш заказ:", reply_markup=types.ReplyKeyboardRemove())
        await bot.send_message(chat_id, '\n'.join(messages))
        json_data = json.dumps(messages)
        add_order(message, json_data)
        await time_order(message)
    else:
        await bot.send_message(chat_id, 'Вы не записали ни одной позиции')
        await NV(message)
```

Рис. 8. Функции ограничений количества позиций

В завершающей части оформления предзаказа бот предложит выбрать время посещения кофейни после заполнения всех позиций пользователем. Это будет последний этап для формирования предзаказа. После бот выведет сообщения со всеми введенными данными, такими как филиал, выбранные позиции и время, через которое пользователь прибудет в кофейню. Если же что-то окажется неверным или пользователь передумал, либо решил выбрать что-то другое, то возможно вернуться к оформлению заказа, нажав на соответствующую кнопку, которая перенаправит к навигационному меню. Если же все указано корректно, то пользователь подтверждает свой заказ и ожидает его приготовление, о котором его должен оповестить бот. В боте предусмотрено 5 состояний заказа:

0 – это черновик, когда пользователь только формирует свой предзаказ;

1 – пользователь отправил свой предзаказ;

2 – предзаказ принят в работу;

3 – предзаказ готов;

-1 – предзаказ приготовить не удалось.

После пользователю придет сообщение о том, что его заказ готов и его можно забрать, но возможно и такое, что по некоторым причинам заказ приготовить не удастся и гость останется без своего заказа. Последний этап оформления заказа и состояния заказа продемонстрированы на рисунке 9.

Рис. 9. Оформление заказа, состояния заказа

Создание телеграм-бота для оформления предзаказов в сети кофеен осуществлялось с использованием языка Python, фреймворка aiogram и БД SQLite представляет интересный пример применения технологий для упрощения процесса заказа и повышения удобства обслуживания клиентов.

Основным инструментом, используемым для создания бота, является язык программирования Python. Python является одним из самых популярных языков для разработки ботов и приложений благодаря своей простоте, гибкости и широкому выбору библиотек [4].

Фреймворк aiogram, выбранный для разработки бота, предоставляет удобные инструменты для работы с Telegram API и обеспечивает простоту создания и поддержки функционала бота [5]. Благодаря этому фреймворку возможно легко настроить обработку команд и сообщений от пользователей, а также реализовать механизм предзаказа.

Использование базы данных SQLite для хранения данных представляется разумным выбором для решения этой задачи. SQLite – это легкое и простое в использовании реляционное хранилище данных, для работы которого не требуется отдельный сервер. Таким образом, SQLite позволяет сохранять информацию о предзаказах без необходимости установки и настройки дополнительных компонентов.

В результате разработки этого Telegram-бота клиенты сети кофеен теперь могут легко размещать предзаказы через мессенджер Telegram, что значительно упрощает процесс оформления заказа и экономит время пользователей. Благодаря использованию Python, фреймворка aiogram и базы данных SQLite разработка бота была относительно простой и эффективной.

Библиографический список

1. Популярные мессенджеры: список российских и мировых сервисов. Обзор 2021. – [Электронный ресурс]. – Режим доступа: https://vc.ru/u/234439-crm-group/298130-populyarnye-messendzhery-spisok-rossiyskih-i-mirovyh-servisov-obzor-2021.
2. Боты для заказа и доставки еды. – [Электронный ресурс]. – Режим доступа: https://www.botobot.ru/blog/ru/boty-dlia-zakaza-i-dostavki-edy/.
3. Статистика Telegram в 2022 году | 100+ фактов. – [Электронный ресурс]. – Режим доступа: https://inclient.ru/telegram-stats/.
4. Как Python стал самым популярным языком программирования. – [Электронный ресурс]. – Режим доступа: https://tproger.ru/articles/pochemu-python-takoj-populjarnyj/.
5. Знакомство с aiogram - Пишем Telegram-ботов с aiogram 3.x (β). – [Электронный ресурс]. – Режим доступа: https://mastergroosha.github.io/aiogram-3-guide/quickstart/.
6. SQLite — самая простая база данных, которая работает везде // Журнал «Код» программирование без снобизма. – [Электронный ресурс]. – Режим доступа: https://thecode.media/sqlite/.

TELEGRAM BOT FOR ORGANIZING PRE-ORDERS IN COFFEE SHOPS

D.P. Pushkareva, *Student*
Supervisor: *L.V. Makukha, Senior Lecturer*
Siberian Federal University
(Russia, Krasnoyarsk)

Abstract. *The article describes the work of a telegram bot for placing orders in a chain of coffee shops of various menu items using the Telegram messenger. The bot includes the selection of a branch, the time of the visit, familiarization with the menu and the formation of a pre-order. The telegram bot greatly simplifies the process of placing orders in a coffee shop, allowing users to place orders and receive information about the status of the order directly through the Telegram messenger. This is convenient for customers and helps the coffee shop to process orders efficiently.*

Keywords: *messenger, telegram bot, bot, Telegram, coffee, coffee shop, order, pre-order.*