КУРСОВАЯ РАБОТА

«Разработка Telegram-бота для обработки заказов»

По дисциплине: «Современные технологии программирования»

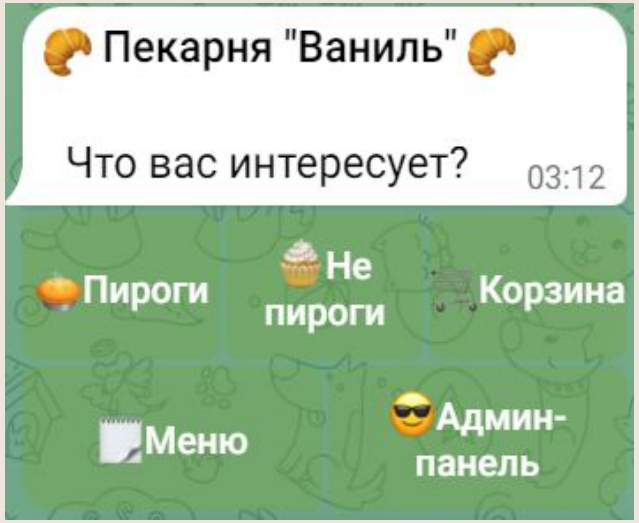
Выполнила :
студент группы ЗБ
Жуковец И.С.
Руководитель: Ягелло А.А.

Цель курсовой работы:

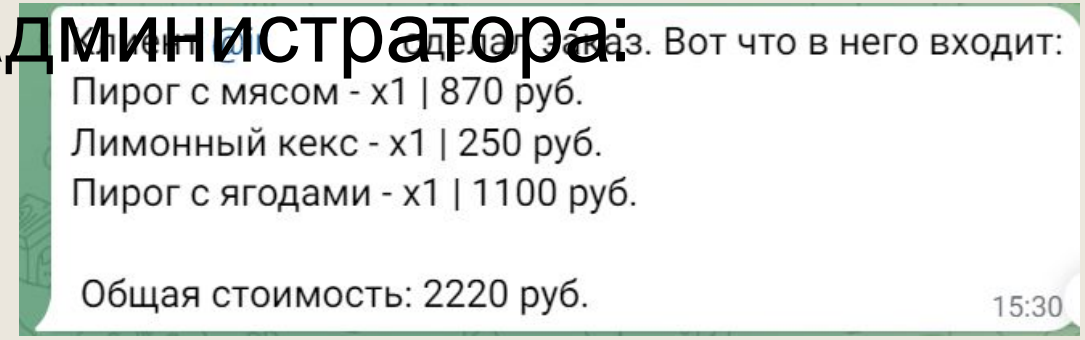
- Разработать Telegram-бот для обработки заказов

Задачи

- Провести анализ предметной области.
- Выбрать среду разработки.
- Выбрать СУБД, провести сравнение возможных вариантов.
- Изучить все варианты существующих библиотек для написания современных Telegram-ботов.
- Разработать и реализовать библиотеки и функции.



Сообщение для администратора:



Анализ предметной области

Преимущества Telegram-ботов:

- Моментальный ответ в любое время.
- Нет необходимости устанавливать другое приложение.
- Интуитивно понятный интерфейс.
- Широкое разнообразие, которое помогает найти бота на любой вкус.

- Минимализм.
- Эффективность, SQLite использует минимальные ресурсы.
- Высокая скорость.
- Модуль sqlite3.

- Удобство: проверка синтаксиса, автодополнение кода.
- Встроенный отладчик.
- Простота использования.
- Интеграция с другими инструментами.

Создание базы данных

Таблица: Products

	id	name	price	is_pirogi
	Фил...	Фильтр	Фил...	Фильтр
1	1	Синнабон	350.0	0
2	2	Лимонный кекс	250.0	0
3	3	Ореховое пирожное	180.0	0
4	4	Пирог с ягодами	1100.0	1
5	5	Пирог с мясом	870.0	1
6	6	Пирог с кетой	1250.0	1
7	32	Пирог с брокколи	900.0	1
8	33	Панкейки	210.0	0

Таблица «Products»

Таблица: Users

	ID	user_id	full_name
	Фил...	Фильтр	Фильтр
1	17	523	Юл.
2	18	462	Ar
3	21	137	Я
4	22	101	am

Таблица «Users»

	ID	user_id	product
	Фил...	Фильтр	Фильтр
1	263	101	Лимонный кекс
2	264	101	Лимонный кекс
3	265	101	Пирог с ягодами
4	269	137	Пирог с ягодами
5	270	137	Пирог с мясом

Таблица «Cart»

```
import asyncio # библиотека позволяющая выполнять функции ассинхронно

import sqlite3 # библиотека базы данных

import logging # логирование работы программы

from aiogram import Bot, Dispatcher, types, F # библиотеки и модули
для работы с телеграм

from aiogram.filters import Command

from aiogram.enums import ParseMode

from aiogram.types import Message, FSInputFile, URLInputFile,
BufferedInputFile

from aiogram.utils.keyboard import InlineKeyboardBuilder

from aiogram.fsm.storage.memory import MemoryStorage

from aiogram.fsm.state import State, StatesGroup

from aiogram.fsm.context import FSMContext

from config import token
```

Импортированные библиотеки

```
@dp.callback_query(F.data.startswith("toCart_"))
async def add_to_cart(callback: types.CallbackQuery):
    action = callback.data.split("_")[1]
    print(action)
    connection = sqlite3.connect('bakery.db')
    cursor = connection.cursor()
    cursor.execute('INSERT INTO Cart (user_id, product) VALUES (?,
?);',
                   (str(callback.from_user.id), str(action)))
    connection.commit()
    connection.close()
    await callback.answer(text=f"{str(action)} добавлен в корзину!")
```

Пример использования хэндлера


```
if message.from_user.id == 137        :
        builder.add(types.InlineKeyboardButton(
            text="😎Админ-панель",
            callback_data="AdminPanel")
        )
```

Если id пользователя совпадает с id администратора – предоставляется доступ к панели администратора

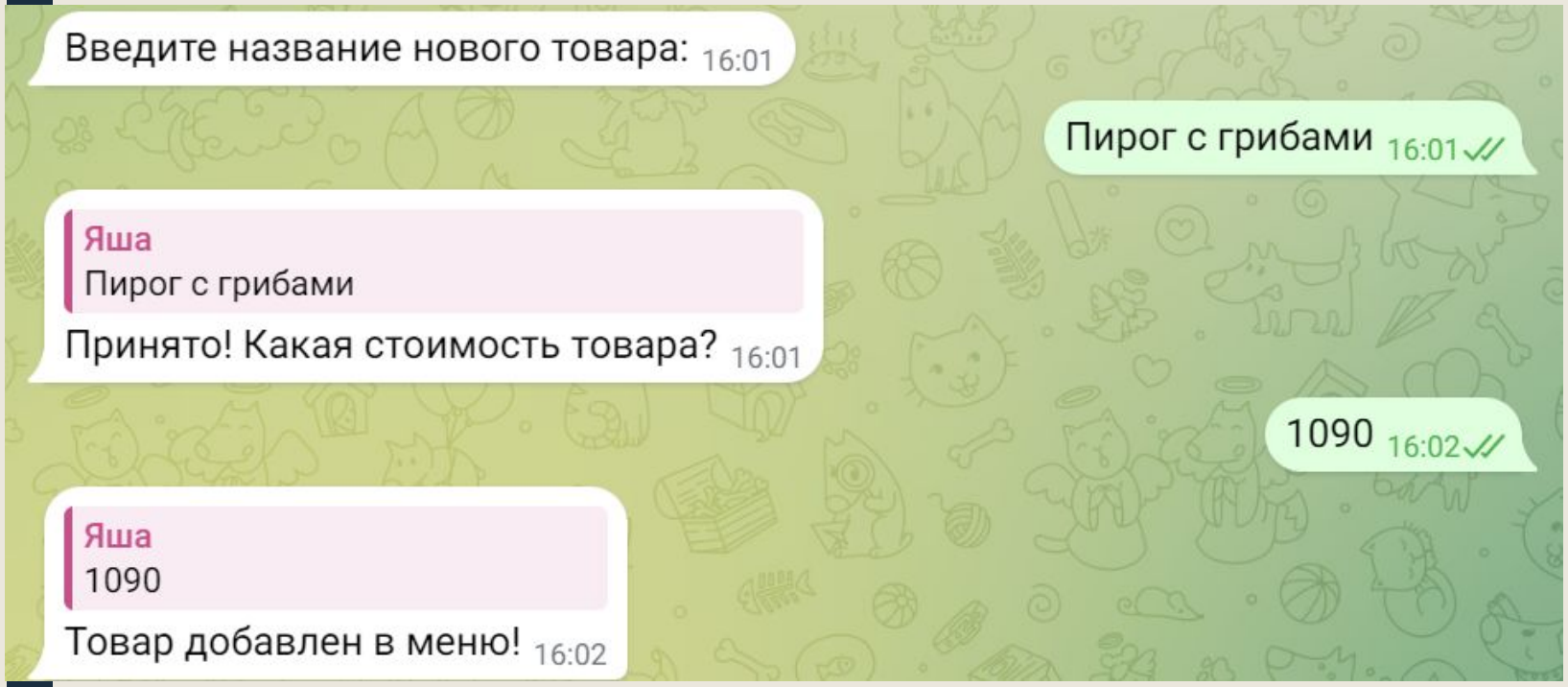
```
@dp.callback_query(F.data == "AdminPanel")
async def admin_panel(callback: types.CallbackQuery):
    if callback.from_user.id == 137       :
        builder = InlineKeyboardBuilder()
        builder.add(types.InlineKeyboardButton(
            text="🫓Пирог",
            callback_data="add_pirog")
        )
...
```

Повторная проверка на доступ к панели администратора

Класс состояний

```
class AdminStates(StatesGroup):


    name = State()
    price = State()
    delete = State()
```



Добавление новых товаров в меню

```
@dp.callback_query(F.data.startswith("add_"))

async def add_to_menu(callback: types.CallbackQuery, state:
FSMContext):

    #user_value = user_data.get(callback.from_user.id, 0)

    action = callback.data.split("_")[1]

    if action == 'pirog':

        await state.update_data(product_status=1)

    else:

        await state.update_data(product_status=0)

    await callback.message.edit_text('Введите название нового
товара:')

    await state.set_state(AdminStates.name) #Установили состояние
ввода имени нового товара
```

```
@dp.message(AdminStates.name) # хэндлер отлова сообщений, но только из
состояния name

async def new_name(message: types.Message, state: FSMContext) -> None:

    await state.update_data(product_name=message.text)

    await message.reply('Принято! Какая стоимость товара?')

    await state.set_state(AdminStates.price) # установили состояние
цены
```

Следующая функция будет вызываться только из состояния price и сохранит полученную информацию, если типы данных

Заключение

- Разработали удобный Telegram-бот для обработки заказов.
- Провели анализ предметной области, отметили преимущества использования ботов.
- Выбрали среду разработки «PyCharm».
- Сравнили популярные СУБД и выбрали «SQLite».
- Реализовали функции и библиотеки.
- Протестировали продукт. Он работает без ошибок и содействует экономии времени и удержанию внимания потенциальных клиентов.

СПАСИБО ЗА ВНИМАНИЕ