Московский государственный технический
университет им. Н.Э. Баумана

Факультет информатика и системы управления
Кафедра системы обработки информации и управления

Курс «Парадигмы и конструкции языков программирования»
Отчет по лабораторной работы №3

Выполнил:
студент группы ИУ5-32Б:
Шпакова Д. Л.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Нардид А.Н.
Подпись и дата:

Москва, 2025 г.

Описание задания

Цель лабораторной работы: изучение разработки ботов в Telegram.

Задание:

1. Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

Текст программы

`main.py`

```
import asyncio
import os
from aiogram import Bot, Dispatcher
from dotenv import load_dotenv
from app.handlers import router
from app.database.models import async_main
load_dotenv()

BOT_TOKEN = os.getenv('BOT_TOKEN')
if not BOT_TOKEN:
    raise ValueError("BOT_TOKEN не найден в файле .env")

async def main():
    await async_main()
    bot = Bot(token= BOT_TOKEN)
    dp = Dispatcher()
    dp.include_router(router)
    await dp.start_polling(bot)

if __name__ == '__main__':
    try:
        asyncio.run(main())
    except KeyboardInterrupt:
        print('Бот выключен')
```

`app/handlers.py`

```
from aiogram import F, Router
from aiogram.types import Message, CallbackQuery
from aiogram.filters import CommandStart, Command
from aiogram.fsm.state import State, StatesGroup
from aiogram.fsm.context import FSMContext

import app.keyboards as kb
import app.database.requests as rq

router = Router()

class Register(StatesGroup):
    name = State()
    age = State()
    number = State()

@router.message(CommandStart())
async def cmd_start(message: Message):
    await rq.set_user(message.from_user.id)
    await rq.clear_cart(message.from_user.id)
```

```
    await message.answer('Добро пожаловать в галерею искусств!',
reply_markup=kb.main)

@router.message(Command('help'))
async def cmd_help(message: Message):
    await message.answer("Это команда /help. Чем я могу помочь?")

@router.message(F.text == 'Каталог')
async def catalog(message: Message):
    await message.answer("Выберите жанр произведения", reply_markup=await
kb.genres())

@router.callback_query(F.data.startswith('genre_'))
async def genre_selected(callback: CallbackQuery):
    genre_id = callback.data.split('_')[1]
    await callback.answer('Вы выбрали жанр')
    await callback.message.edit_text('Выберите произведение',
                                     reply_markup=await kb.arts(genre_id))

@router.callback_query(F.data.startswith('art_'))
async def art_selected(callback: CallbackQuery):
    art_id = callback.data.split('_')[1]
    art_data = await rq.get_art(art_id)
    await callback.answer('Вы выбрали картину')

    await callback.message.edit_text(
        f'Название: "{art_data.name}"\n'
        f'Художник: {art_data.artist}\n'
        f'Цена: {art_data.price} $',
        reply_markup=kb.art_actions(art_id)
          )

@router.message(Command('register'))
async def register(message: Message, state: FSMContext):
    await state.set_state(Register.name)
    await message.answer('Введите ваше имя')

@router.message(Register.name)
async def register_name(message: Message, state: FSMContext):
    await state.update_data(name=message.text)
    await state.set_state(Register.age)
    await message.answer('Введите ваш возраст')

@router.message(Register.age)
async def register_age(message: Message, state: FSMContext):
    await state.update_data(age=message.text)
    await state.set_state(Register.number)
    await message.answer('Отправьте ваш номер телефона',
reply_markup=kb.get_number)

@router.callback_query(F.data.startswith('art_'))
```

```python
async def art_selected(callback: CallbackQuery):
    art_id = callback.data.split('_')[1]
    art_data = await rq.get_art(art_id)

    await callback.message.edit_text(
        f'Название: "{art_data.name}"\n'
        f'Художник: {art_data.artist}\n'
        f'Цена: {art_data.price} $',
        reply_markup=kb.art_actions(art_id) # <--- Прикрепляем новую клавиатуру
    )

@router.callback_query(F.data.startswith('add_cart_'))
async def add_to_cart_handler(callback: CallbackQuery):
    art_id = callback.data.split('_')[2]
    await rq.add_to_cart(callback.from_user.id, art_id)

    await callback.answer('Товар добавлен в корзину!')

@router.message(F.text == 'Корзина')
async def show_cart(message: Message):

    cart_items = await rq.get_cart(message.from_user.id)


    items_list = list(cart_items)
    if not items_list:
        await message.answer("Ваша корзина пуста 🗑")
        return

    await message.answer("🛒 <b>Ваша корзина:</b>", parse_mode='HTML')

    total_price = 0
    for item in items_list:
        await message.answer(f'🖼 <b>{item.name}</b>\nАвтор: {item.artist}\nЦена:
{item.price} $', parse_mode='HTML')
        total_price += item.price

    await message.answer(f'💰 <b>Итого: {total_price} $</b>', parse_mode='HTML')

@router.message(F.text == 'О нас')
async def about_us(message: Message):
    about_text = (
        "👋 Добро пожаловать в Арт-галерею!\n\n"
        "Я — Арт-Бот, ваш персональный проводник в мире искусства.\n\n"
        "Моя задача — помочь вам легко найти, просмотреть и выбрать произведение,
"
        "которое вам понравится.\n\n"
        "Надеюсь, вам будет комфортно и интересно в моей галерее! 🖼"
    )
    await message.answer(about_text)
```

`app/keyboards.py`

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton,
InlineKeyboardButton
from aiogram.utils.keyboard import InlineKeyboardBuilder
from app.database.requests import get_genres, get_genre_arts

main = ReplyKeyboardMarkup(keyboard=[[KeyboardButton(text='Каталог')],
                                     [KeyboardButton(text='Корзина')],
                                     [KeyboardButton(text='О нас')]],
                           resize_keyboard=True,
                           input_field_placeholder='Выберите пункт меню')

get_number = ReplyKeyboardMarkup(keyboard=[[KeyboardButton(text='Отправить
номер', request_contact=True)]],
                                 resize_keyboard=True)

async def genres():
    all_genres = await get_genres()
    keyboard = InlineKeyboardBuilder()
    for genre in all_genres:
        keyboard.add(InlineKeyboardButton(text=genre.name,
callback_data=f"genre_{genre.id}"))
    keyboard.add(InlineKeyboardButton(text='На главную',
callback_data='to_main'))
    return keyboard.adjust(2).as_markup()

async def arts(genre_id):
    all_arts = await get_genre_arts(genre_id)
    keyboard = InlineKeyboardBuilder()
    for art in all_arts:
        keyboard.add(InlineKeyboardButton(text=art.name,
callback_data=f"art_{art.id}"))
    keyboard.add(InlineKeyboardButton(text='На главную',
callback_data='to_main'))
    return keyboard.adjust(2).as_markup()

def art_actions(art_id):
    keyboard = InlineKeyboardBuilder()
    keyboard.add(InlineKeyboardButton(text='Добавить в корзину',
callback_data=f'add_cart_{art_id}'))
    keyboard.add(InlineKeyboardButton(text='На главную',
callback_data='to_main'))
    return keyboard.adjust(1).as_markup()
```

`app/database/models.py`

```
from sqlalchemy import BigInteger, String, ForeignKey
from sqlalchemy.orm import DeclarativeBase, Mapped, mapped_column
from sqlalchemy.ext.asyncio import AsyncAttrs, async_sessionmaker,
create_async_engine
```

```python
engine = create_async_engine(url='sqlite+aiosqlite:///db.sqlite3')

async_session = async_sessionmaker(engine)

class Base(AsyncAttrs, DeclarativeBase):
    pass

class User(Base):
    __tablename__ = 'users'
    id: Mapped[int] = mapped_column(primary_key=True)
    tg_id = mapped_column(BigInteger)

class Genre(Base):
    __tablename__ = 'genres'
    id: Mapped[int] = mapped_column(primary_key=True)
    name: Mapped[str] = mapped_column(String(25))

class Art(Base):
    __tablename__ = 'arts'
    id: Mapped[int] = mapped_column(primary_key=True)
    name: Mapped[str] = mapped_column(String(25))
    artist: Mapped[str] = mapped_column(String(120))
    price: Mapped[int] = mapped_column()
    genre: Mapped[int] = mapped_column(ForeignKey('genres.id'))

class Cart(Base):
    __tablename__ = 'cart'
    id: Mapped[int] = mapped_column(primary_key=True)
    user_tg_id: Mapped[int] = mapped_column(BigInteger)
    art_id: Mapped[int] = mapped_column(ForeignKey('arts.id'))

async def async_main():
    async with engine.begin() as conn:
        await conn.run_sync(Base.metadata.create_all)
```

`app/database/requests.py`

```python
from app.database.models import async_session
from app.database.models import User, Genre, Art, Cart
from sqlalchemy import select, delete

async def set_user(tg_id):
    async with async_session() as session:
        user = await session.scalar(select(User).where(User.tg_id == tg_id))

        if not user:
            session.add(User(tg_id=tg_id))
            await session.commit()

async def get_genres():
```

```
    async with async_session() as session:
        return await session.scalars(select(Genre))

async def get_genre_arts(genre_id):
    async with async_session() as session:
        return await session.scalars(select(Art).where(Art.genre == genre_id))

async def get_art(art_id):
    async with async_session() as session:
        return await session.scalar(select(Art).where(Art.id == art_id))

async def add_to_cart(tg_id, art_id):
    async with async_session() as session:
        session.add(Cart(user_tg_id=tg_id, art_id=art_id))
        await session.commit()

async def get_cart(tg_id):
    async with async_session() as session:
        return await session.scalars(select(Art).join(Cart).where(Cart.user_tg_id
== tg_id))

async def clear_cart(tg_id):
    async with async_session() as session:
        await session.execute(delete(Cart).where(Cart.user_tg_id == tg_id))
        await session.commit()
```

Скриншот работы приложения

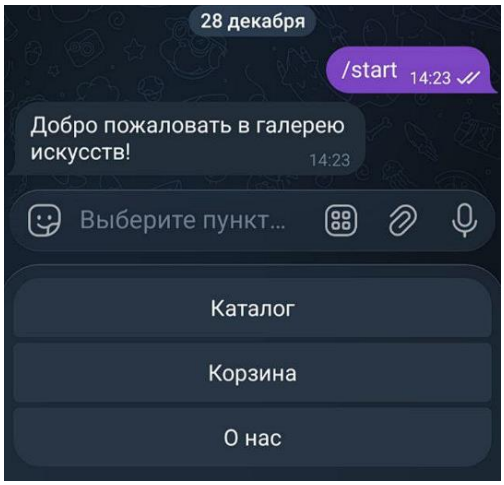

Рис. 1 Ответ на сообщение /start

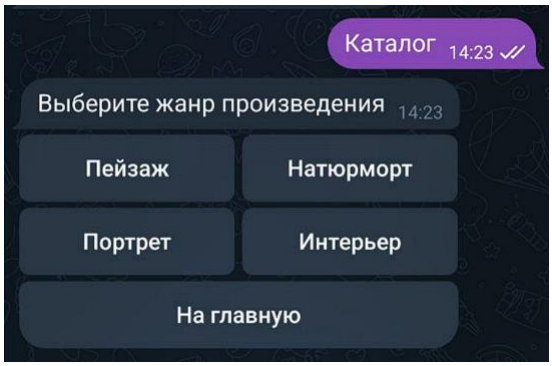

Рис. 2 Ответ на сообщение Каталог

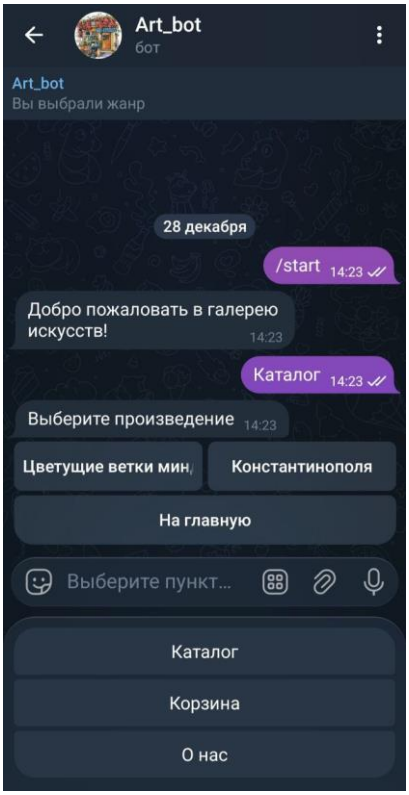

Рис. 3 Выюор жанра Пейзаж

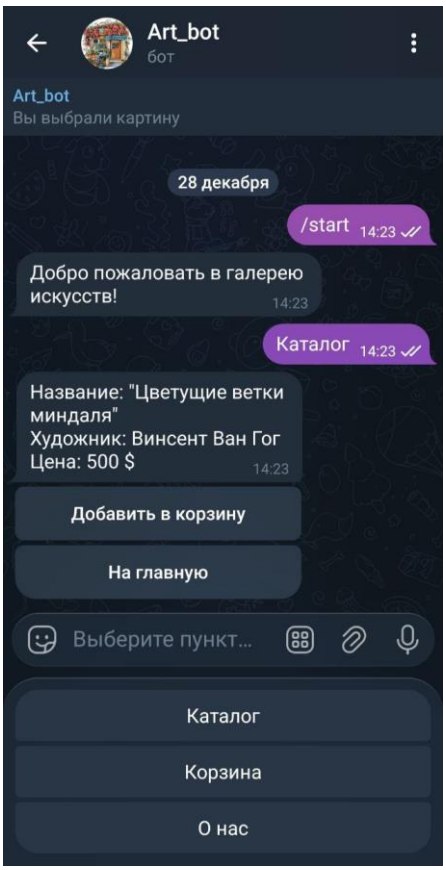

Рис. 4 Выбор картины

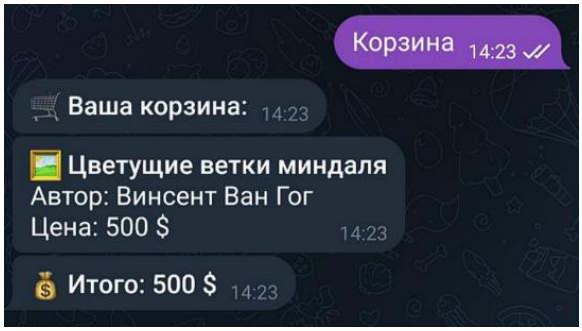

Рис. 5 Просмотр корзины

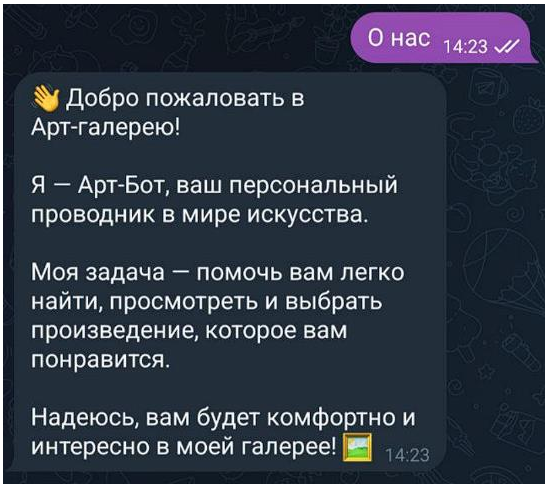

Рис. 6 Ответ на сообщение О нас