```
import asyncio
import sqlite3
import random
from aiogram import Bot, Dispatcher, types
from aiogram.filters import Command
from aiogram.types import
ReplyKeyboardMarkup, KeyboardButton,
InlineKeyboardMarkup, InlineKeyboardButton

# Токен бота
TOKEN = "ВАШ_ТОКЕН"

bot = Bot(token=TOKEN)
dp = Dispatcher()

# Подключение к базе данных
conn = sqlite3.connect("shopping_list.db")
cursor = conn×cursor()

# Таблицы для списков
cursor.execute("""
CREATE TABLE IF NOT EXISTS lists (
    id INTEGER PRIMARY KEY,
    name TEXT,
    owner_id INTEGER
)
""")
cursor.execute("""
CREATE TABLE IF NOT EXISTS users_lists (
    user_id INTEGER,
    list_id INTEGER
)
""")
cursor.execute("""
CREATE TABLE IF NOT EXISTS shopping_list (
    id INTEGER PRIMARY KEY
AUTOINCREMENT,
    list_id INTEGER,
    item TEXT,
    bought INTEGER DEFAULT 0
)
""")
conn.commit()

# Главное меню
main_menu =
```

```
ReplyKeyboardMarkup(keyboard=[
    [KeyboardButton(text="📋 Мои списки")],
    [KeyboardButton(text="➕ Новый
список"), KeyboardButton(text="📤
Поделиться списком")]
], resize_keyboard=True)

# Создание нового списка
@dp.message(Command("new_list"))
async def create_list(message:
types.Message):
    list_id = random.randint(1000, 9999)
    cursor.execute("INSERT INTO lists (id,
name, owner_id) VALUES (?, ?, ?)", (list_id,
"Мой список", message.from_user.id))
    cursor.execute("INSERT INTO users_lists
(user_id, list_id) VALUES (?, ?)",
(message.from_user.id, list_id))
    conn.commit()
    await message.answer(f"✅ Создан новый
список (ID: {list_id})!")

# Просмотр своих списков
@dp.message(Command("my_lists"))
async def my_lists(message: types.Message):
    user_id = message.from_user.id
    cursor.execute("SELECT lists.id, lists.name
FROM lists JOIN users_lists ON lists.id =
users_lists.list_id WHERE
users_lists.user_id=?", (user_id,))
    lists = cursor×fetchall()

    if not lists:
        await message.answer("📋 У тебя нет
списков.")
    else:
        response = "📋 Твои списки:\n"
        for list_id, name in lists:
            response += f"🔷 {name} (ID: {list_id})
\n"
        await message.answer(response)

# Поделиться списком
@dp.message(Command("share"))
async def share_list(message:
types.Message):
```

```
    user_id = message.from_user.id
    cursor.execute("SELECT id FROM lists
WHERE owner_id=?", (user_id,))
    lists = cursor×fetchall()

    if not lists:
        await message.answer("📤 У тебя нет
списков для отправки.")
    else:
        response = "📤 Поделись этими
ссылками:\n"
        for list_id, in lists:
            response += f"🔗 https://t.me/
ТВОЙ_БОТ?start=join_{list_id}\n"
        await message.answer(response)

# Присоединение к списку
@dp.message(lambda message:
message.text.startswith("/join_"))
async def join_list(message: types.Message):
    try:
        list_id = int(message.text.split("_")[1])
        user_id = message.from_user.id
        cursor.execute("INSERT INTO users_lists
(user_id, list_id) VALUES (?, ?)", (user_id,
list_id))
        conn.commit()
        await message.answer(f"✅ Теперь у
тебя есть доступ к списку {list_id}!")
    except:
        await message.answer("❌ Ошибка!
Проверь команду.")

# Добавление продукта
@dp.message(Command("add"))
async def add_item(message:
types.Message):
    user_id = message.from_user.id
    cursor.execute("SELECT list_id FROM
users_lists WHERE user_id=? LIMIT 1",
(user_id,))
    list_id = cursor×fetchone()

    if not list_id:
        await message.answer("❌ У тебя нет
списка! Создай новый с /new_list")
```

```
        return

    list_id = list_id[0]
    item = message.text.replace("/add ",
"").strip()

    if item:
        cursor.execute("INSERT INTO
shopping_list (list_id, item) VALUES (?, ?)",
(list_id, item))
        conn.commit()
        await message.answer(f"✅ {item}
добавлен в список!")
    else:
        await message.answer("❌ Напиши
продукт после команды /add")

# Просмотр списка с возможностью
отметить покупки
@dp.message(Command("view"))
async def view_list(message:
types.Message):
    user_id = message.from_user.id
    cursor.execute("SELECT list_id FROM
users_lists WHERE user_id=? LIMIT 1",
(user_id,))
    list_id = cursor×fetchone()

    if not list_id:
        await message.answer("📋 У тебя нет
списка! Создай новый с /new_list")
        return

    list_id = list_id[0]
    cursor.execute("SELECT id, item, bought
FROM shopping_list WHERE list_id=?",
(list_id,))
    items = cursor×fetchall()

    if not items:
        await message.answer("🛒 Твой список
пуст!")
    else:
        keyboard = InlineKeyboardMarkup()
        for item_id, item, bought in items:
            status = "✅ " if bought else "⬜ "
```

```
keyboard×add(InlineKeyboardButton(text=f"{
status}{item}",
callback_data=f"toggle_{item_id}"))

    await message.answer("📋 Твой список
покупок:", reply_markup=keyboard)

# Отметить купленный товар
@dp.callback_query(lambda call:
call.data.startswith("toggle_"))
async def toggle_item(call:
types.CallbackQuery):
    item_id = int(call×data×split("_")[1])

    cursor.execute("SELECT bought FROM
shopping_list WHERE id=?", (item_id,))
    bought = cursor×fetchone()[0]

    new_status = 0 if bought else 1
    cursor.execute("UPDATE shopping_list SET
bought=? WHERE id=?", (new_status,
item_id))
    conn.commit()

    await call.answer("Статус обновлён!")
    await view_list(call.message)

# Очистить список
@dp.message(Command("clear"))
async def clear_list(message:
types.Message):
    user_id = message.from_user.id
    cursor.execute("SELECT list_id FROM
users_lists WHERE user_id=? LIMIT 1",
(user_id,))
    list_id = cursor×fetchone()

    if not list_id:
        await message.answer("🗑️ У тебя нет
списка!")
        return

    list_id = list_id[0]
    cursor.execute("DELETE FROM
shopping_list WHERE list_id=?", (list_id,))
```

```
    conn.commit()
    await message.answer("🗑️ Список
покупок очищен!")

# Запуск бота
async def main():
    await dp.start_polling(bot)

if __name__ == "__main__":
    asyncio.run(main())
```