Теоретические материалы к занятию 7

С ограничением доступа мы уже немного знакомы, но прежде чем вплотную этим заняться давайте допишем нашего бота. Теперь перейдем в папку user и создадим файл cart для корзины

```
@dp.message_handler(IsUser(), text=cart)
async def process_cart(message: Message, state: FSMContext):

   cart_data = db.fetchall(
       'SELECT * FROM cart WHERE cid=?', (message.chat.id,))

   if len(cart_data) == 0:

       await message.answer('Ваша корзина пуста.')

   else:

       await bot.send_chat_action(message.chat.id,
ChatActions.TYPING)
       async with state.proxy() as data:
           data['products'] = {}

       order_cost = 0

       for _, idx, count_in_cart in cart_data:

           product = db.fetchone('SELECT * FROM products WHERE
idx=?', (idx,))

           if product == None:

               db.query('DELETE FROM cart WHERE idx=?', (idx,))

           else:
               _, title, body, image, price, _ = product
               order_cost += price

               async with state.proxy() as data:
                   data['products'][idx] = [title, price,
count_in_cart]

               markup = product_markup(idx, count_in_cart)
               text = f'<b>{title}</b>\n\n{body}\n\nЦена:
{price}₽.'

               await message.answer_photo(photo=image,
                                          caption=text,
                                          reply_markup=markup)

       if order_cost != 0:
```

```
            markup = ReplyKeyboardMarkup(resize_keyboard=True,
selective=True)
            markup.add('📦 Оформить заказ')

            await message.answer('Перейти к оформлению?',
                                 reply_markup=markup)


@dp.callback_query_handler(IsUser(),
product_cb.filter(action='count'))
@dp.callback_query_handler(IsUser(),
product_cb.filter(action='increase'))
@dp.callback_query_handler(IsUser(),
product_cb.filter(action='decrease'))
async def product_callback_handler(query: CallbackQuery,
callback_data: dict, state: FSMContext):

   idx = callback_data['id']
   action = callback_data['action']

   if 'count' == action:

       async with state.proxy() as data:

           if 'products' not in data.keys():

               await process_cart(query.message, state)

           else:

               await query.answer('Количество - ' +
data['products'][idx][2])

   else:

       async with state.proxy() as data:

           if 'products' not in data.keys():

               await process_cart(query.message, state)

           else:

               data['products'][idx][2] += 1 if 'increase' ==
action else -1
               count_in_cart = data['products'][idx][2]

               if count_in_cart == 0:

                   db.query('''DELETE FROM cart
                   WHERE cid = ? AND idx = ?''',
(query.message.chat.id, idx))

                   await query.message.delete()
               else:
```

```
                    db.query('''UPDATE cart
                    SET quantity = ?
                    WHERE cid = ? AND idx = ?''', (count_in_cart,
query.message.chat.id, idx))

                    await
query.message.edit_reply_markup(product_markup(idx,
count_in_cart))


@dp.message_handler(IsUser(), text='📦 Оформить заказ')
async def process_checkout(message: Message, state: FSMContext):

   await CheckoutState.check_cart.set()
   await checkout(message, state)


async def checkout(message, state):
   answer = ''
   total_price = 0

   async with state.proxy() as data:

       for title, price, count_in_cart in
data['products'].values():

           tp = count_in_cart * price
           answer += f'<b>{title}</b> * {count_in_cart}шт. =
{tp}₽\n'
           total_price += tp

   await message.answer(f'{answer}\nОбщая сумма заказа:
{total_price}₽.',
                        reply_markup=check_markup())


@dp.message_handler(IsUser(), lambda message: message.text not in
[all_right_message, back_message],
state=CheckoutState.check_cart)
async def process_check_cart_invalid(message: Message):
   await message.reply('Такого варианта не было.')


@dp.message_handler(IsUser(), text=back_message,
state=CheckoutState.check_cart)
async def process_check_cart_back(message: Message, state:
FSMContext):
   await state.finish()
   await process_cart(message, state)


@dp.message_handler(IsUser(), text=all_right_message,
state=CheckoutState.check_cart)
```

```
async def process_check_cart_all_right(message: Message, state:
FSMContext):
   await CheckoutState.next()
   await message.answer('Укажите свое имя.',
                        reply_markup=back_markup())


@dp.message_handler(IsUser(), text=back_message,
state=CheckoutState.name)
async def process_name_back(message: Message, state: FSMContext):
   await CheckoutState.check_cart.set()
   await checkout(message, state)


@dp.message_handler(IsUser(), state=CheckoutState.name)
async def process_name(message: Message, state: FSMContext):

   async with state.proxy() as data:

       data['name'] = message.text

       if 'address' in data.keys():

           await confirm(message)
           await CheckoutState.confirm.set()

       else:

           await CheckoutState.next()
           await message.answer('Укажите свой адрес места
жительства.',
                                reply_markup=back_markup())


@dp.message_handler(IsUser(), text=back_message,
state=CheckoutState.address)
async def process_address_back(message: Message, state:
FSMContext):

   async with state.proxy() as data:

       await message.answer('Изменить имя с <b>' + data['name'] +
'</b>?',
                            reply_markup=back_markup())

   await CheckoutState.name.set()


@dp.message_handler(IsUser(), state=CheckoutState.address)
async def process_address(message: Message, state: FSMContext):

   async with state.proxy() as data:
       data['address'] = message.text

   await confirm(message)
```

```
   await CheckoutState.next()


async def confirm(message):

   await message.answer('Убедитесь, что все правильно оформлено и
подтвердите заказ.',
                        reply_markup=confirm_markup())


@dp.message_handler(IsUser(), lambda message: message.text not in
[confirm_message, back_message], state=CheckoutState.confirm)
async def process_confirm_invalid(message: Message):
   await message.reply('Такого варианта не было.')


@dp.message_handler(IsUser(), text=back_message,
state=CheckoutState.confirm)
async def process_confirm(message: Message, state: FSMContext):

   await CheckoutState.address.set()

   async with state.proxy() as data:
       await message.answer('Изменить адрес с <b>' +
data['address'] + '</b>?',
                            reply_markup=back_markup())


@dp.message_handler(IsUser(), text=confirm_message,
state=CheckoutState.confirm)
async def process_confirm(message: Message, state: FSMContext):

   enough_money = True  # enough money on the balance sheet
   markup = ReplyKeyboardRemove()

   if enough_money:

       logging.info('Deal was made.')

       async with state.proxy() as data:

           cid = message.chat.id
           products = [idx + '=' + str(quantity)
                       for idx, quantity in db.fetchall('''SELECT
idx, quantity FROM cart
           WHERE cid=?''', (cid,))]  # idx=quantity

           db.query('INSERT INTO orders VALUES (?, ?, ?, ?)',
                    (cid, data['name'], data['address'], '
'.join(products)))

           db.query('DELETE FROM cart WHERE cid=?', (cid,))
```

```
            await message.answer('Ок! Ваш заказ уже в пути
🚀\nИмя: <b>' + data['name'] + '</b>\nАдрес: <b>' +
data['address'] + '</b>',
                                 reply_markup=markup)
    else:

        await message.answer('У вас недостаточно денег на счете.
Пополните баланс!',
                             reply_markup=markup)

    await state.finish()
```

файл catalog

```
@dp.message_handler(IsUser(), text=catalog)
async def process_catalog(message: Message):
    await message.answer('Выберите раздел, чтобы вывести список
товаров:',
                         reply_markup=categories_markup())


@dp.callback_query_handler(IsUser(),
category_cb.filter(action='view'))
async def category_callback_handler(query: CallbackQuery,
callback_data: dict):
    products = db.fetchall('''SELECT * FROM products product
    WHERE product.tag = (SELECT title FROM categories WHERE idx=?)
    AND product.idx NOT IN (SELECT idx FROM cart WHERE cid =
?)''',
                           (callback_data['id'],
query.message.chat.id))

    await query.answer('Все доступные товары.')
    await show_products(query.message, products)


@dp.callback_query_handler(IsUser(),
product_cb.filter(action='add'))
async def add_product_callback_handler(query: CallbackQuery,
callback_data: dict):
    db.query('INSERT INTO cart VALUES (?, ?, 1)',
             (query.message.chat.id, callback_data['id']))

    await query.answer('Товар добавлен в корзину!')
    await query.message.delete()


async def show_products(m, products):
    if len(products) == 0:

        await m.answer('Здесь ничего нет 😢')
```

```
   else:

       await bot.send_chat_action(m.chat.id, ChatActions.TYPING)

       for idx, title, body, image, price, _ in products:
           markup = product_markup(idx, price)
           text = f'<b>{title}</b>\n\n{body}'

           await m.answer_photo(photo=image,
                                caption=text,
                                reply_markup=markup)
```

файл delivery-status

```
@dp.message_handler(IsUser(), text=delivery_status)
async def process_delivery_status(message: Message):
   orders = db.fetchall('SELECT * FROM orders WHERE cid=?',
(message.chat.id,))

   if len(orders) == 0:
       await message.answer('У вас нет активных заказов.')
   else:
       await delivery_status_answer(message, orders)


async def delivery_status_answer(message, orders):
   res = ''

   for order in orders:
       res += f'Заказ <b>№{order[3]}</b>'
       answer = [
           ' лежит на складе.',
           ' уже в пути!',
           ' прибыл и ждет вас на почте!'
       ]

       res += answer[0]
       res += '\n\n'

   await message.answer(res)
```

menu

```
from aiogram.types import Message, CallbackQuery,
ReplyKeyboardMarkup
from loader import dp
from filters import IsAdmin, IsUser
```

```
catalog = '🛍️ Каталог'
balance = '💰 Баланс'
cart = '🛒 Корзина'
delivery_status = '🚚 Статус заказа'

settings = '⚙️ Настройка каталога'
orders = '🚚 Заказы'
questions = '❓ Вопросы'

@dp.message_handler(IsAdmin(), commands='menu')
async def admin_menu(message: Message):
   markup = ReplyKeyboardMarkup(selective=True)
   markup.add(settings)
   markup.add(questions, orders)

   await message.answer('Меню', reply_markup=markup)

@dp.message_handler(IsUser(), commands='menu')
async def user_menu(message: Message):
   markup = ReplyKeyboardMarkup(selective=True)
   markup.add(catalog)
   markup.add(balance, cart)
   markup.add(delivery_status)

   await message.answer('Меню', reply_markup=markup)
```

SOS

```
@dp.message_handler(commands='sos')
async def cmd_sos(message: Message):
   await SosState.question.set()
   await message.answer('В чем суть проблемы? Опишите как можно
детальнее и администратор обязательно вам ответит.',
reply_markup=ReplyKeyboardRemove())


@dp.message_handler(state=SosState.question)
async def process_question(message: Message, state: FSMContext):
   async with state.proxy() as data:
       data['question'] = message.text

   await message.answer('Убедитесь, что все верно.',
reply_markup=submit_markup())
   await SosState.next()


@dp.message_handler(lambda message: message.text not in
[cancel_message, all_right_message], state=SosState.submit)
async def process_price_invalid(message: Message):
   await message.answer('Такого варианта не было.')
```

```
@dp.message_handler(text=cancel_message, state=SosState.submit)
async def process_cancel(message: Message, state: FSMContext):
   await message.answer('Отменено!',
reply_markup=ReplyKeyboardRemove())
   await state.finish()


@dp.message_handler(text=all_right_message,
state=SosState.submit)
async def process_submit(message: Message, state: FSMContext):

   cid = message.chat.id

   if db.fetchone('SELECT * FROM questions WHERE cid=?', (cid,))
== None:

       async with state.proxy() as data:
           db.query('INSERT INTO questions VALUES (?, ?)',
                    (cid, data['question']))

       await message.answer('Отправлено!',
reply_markup=ReplyKeyboardRemove())

   else:

       await message.answer('Превышен лимит на количество
задаваемых вопросов.', reply_markup=ReplyKeyboardRemove())

   await state.finish()
```

wallet

```
@dp.message_handler(IsUser(), text=balance)
async def process_balance(message: Message, state: FSMContext):
   await message.answer('Ваш кошелек пуст! Чтобы его пополнить
нужно...')
```

Ну и добавим стартовые обработчики в основной файл

```
@dp.message_handler(commands='start')
async def cmd_start(message: types.Message):
   markup = ReplyKeyboardMarkup(resize_keyboard=True)

   markup.row(user_message, admin_message)

   await message.answer('''Привет! 👋
🤖 Я бот-магазин по подаже товаров любой категории.
```

```
🛍️ Чтобы перейти в каталог и выбрать приглянувшиеся товары
воспользуйтесь командой /menu.
💰 Пополнить счет можно через Яндекс.кассу, Сбербанк или Qiwi.
❓ Возникли вопросы? Не проблема! Команда /sos поможет связаться
с админами, которые постараются как можно быстрее откликнуться.
🤝 Заказать похожего бота? Свяжитесь с разработчиком <a
href="#">Имя Фамилия</a>
    ''', reply_markup=markup)


@dp.message_handler(text=user_message)
async def user_mode(message: types.Message):
    cid = message.chat.id
    if cid in config.ADMINS:
        config.ADMINS.remove(cid)

    await message.answer('Включен пользовательский режим.',
reply_markup=ReplyKeyboardRemove())


@dp.message_handler(text=admin_message)
async def admin_mode(message: types.Message):
    cid = message.chat.id
    if cid not in config.ADMINS:
        config.ADMINS.append(cid)

    await message.answer('Включен адмнский режим.',
reply_markup=ReplyKeyboardRemove())
```

Бот готов!