Мой обработчик переведён в состояние await state.set\_state(Order\_all.vibor\_meals) но выбираемая команда записана как /53068 и программа выдаёт ошибку TypeError: vivod\_meals() missing 1 required positional argument: 'command' . Как правильно исправить программу?

ChatGPT

Ошибку TypeError: vivod\_meals() missing 1 required positional argument: 'command' можно исправить, изменив обработчик так, чтобы он не требовал аргумента command, или обеспечив правильную передачу этого аргумента.

Если команда /53068 должна обрабатываться в состоянии Order\_all.vibor\_meals, нужно убедиться, что обработчик состояния правильно принимает аргументы.

Вот пример того, как можно правильно настроить обработчик состояния для команды /53068:

python

Copy code

from aiogram import Bot, Dispatcher, types, Router

from aiogram.fsm.context import FSMContext

from aiogram.filters import CommandStart, Command

from aiogram.fsm.state import State, StatesGroup

from aiogram.types import Message

# Ваши токены и настройки

TOKEN = "YOUR\_BOT\_TOKEN"

bot = Bot(token=TOKEN)

dp = Dispatcher()

# Создание группы состояний

class Order\_all(StatesGroup):

waiting\_meals = State()

vibor\_meals = State()

vivod\_meals = State()

router = Router()

# Обработчик для состояния waiting\_meals

@router.message(Order\_all.waiting\_meals)

async def waiting\_meals\_handler(message: Message, state: FSMContext) -> None:

# Ваша логика

await message.answer("Вы в состоянии waiting\_meals. Переключаюсь на vibor\_meals.")

await state.set\_state(Order\_all.vibor\_meals)

# Обработчик для состояния vibor\_meals

@router.message(Order\_all.vibor\_meals, Command())

async def vibor\_meals\_handler(message: Message, command: CommandObject, state: FSMContext) -> None:

# Ваша логика

if command.command == "/53068":

await message.answer("Вы выбрали команду /53068. Переключаюсь на vivod\_meals.")

await state.set\_state(Order\_all.vivod\_meals)

# Обработчик для состояния vivod\_meals

@router.message(Order\_all.vivod\_meals)

async def vivod\_meals\_handler(message: Message, state: FSMContext) -> None:

# Ваша логика

await message.answer("Теперь вы в состоянии vivod\_meals.")

# Обработчик для команды /start

@router.message(CommandStart())

async def command\_start\_handler(message: Message, state: FSMContext) -> None:

await message.answer("Привет! Начинаем заказ.")

await state.set\_state(Order\_all.waiting\_meals)

# Регистрация обработчиков

dp.include\_router(router)

# Запуск бота

async def main():

await dp.start\_polling(bot)

if \_\_name\_\_ == "\_\_main\_\_":

import asyncio

asyncio.run(main())

В этом примере:

1. Обработчик состояния waiting\_meals переводит состояние в vibor\_meals.
2. Обработчик состояния vibor\_meals проверяет, является ли команда /53068, и если да, переводит состояние в vivod\_meals.
3. Обработчик состояния vivod\_meals отвечает на сообщения в этом состоянии.

Если у вас другие команды, их нужно аналогично обрабатывать в соответствующих состояниях.

Пример использования FSMContext для сохранения динамического списка команд:

python

Copy code

from aiogram import Router, types

from aiogram.filters import Command

from aiogram.fsm.context import FSMContext

from aiogram.fsm.state import StatesGroup, State

from aiogram.fsm.storage.memory import MemoryStorage

class Order\_all(StatesGroup):

vibor\_meals = State()

router = Router()

# Пример предыдущего шага, где формируется список команд

@router.message(Command(commands=["set\_commands"]))

async def set\_commands(message: types.Message, state: FSMContext):

# Пример динамического списка команд

commands\_list = ["start", "help", "order"]

await state.update\_data(commands\_list=commands\_list)

await Order\_all.vibor\_meals.set()

await message.answer("Commands set. Proceed with your action.")

# Обработчик сообщения, который фильтрует команды

@router.message(Order\_all.vibor\_meals)

async def handle\_dynamic\_commands(message: types.Message, state: FSMContext):

data = await state.get\_data()

commands\_list = data.get("commands\_list", [])

command = message.text.strip("/").split()[0] # Извлекаем команду из текста сообщения

if command in commands\_list:

await message.answer(f"Handling command: {command}")

else:

await message.answer(f"Unknown command: {command}")

# Настройка FSM Storage

storage = MemoryStorage()

# Создание экземпляра Dispatcher с настроенной storage

from aiogram import Dispatcher

dp = Dispatcher(storage=storage)

# Регистрация роутеров

dp.include\_router(router)

# Запуск бота

if \_\_name\_\_ == "\_\_main\_\_":

from aiogram import Bot, executor

bot = Bot(token="YOUR\_BOT\_TOKEN\_HERE")

executor.start\_polling(dp, skip\_updates=True)

В этом примере:

1. На первом шаге создается динамический список команд и сохраняется в состоянии пользователя.