from aiogram.utils.helper import Helper, HelperMode, ListItem  
from aiogram import Dispatcher, types  
from aiogram.dispatcher import FSMContext  
from aiogram.dispatcher.filters import state  
from aiogram.types.base import String  
from aiogram.utils import executor, callback\_data  
from create\_bot import dp, bot  
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton, ContentType  
import time  
from aiogram.dispatcher.filters.state import State, StatesGroup  
import asyncio  
from apscheduler.schedulers.asyncio import AsyncIOScheduler  
from handlers import apsched, rules, kick\_user, price\_checker  
from database import database\_conn  
  
PAYMENTS\_TOKEN = '401643678:TEST:d009db99-1e8f-454e-ae3a-dba629e886d5'  
b1 = KeyboardButton('/Оплатить', *callback\_data*='user\_id')  
b2 = KeyboardButton('/Меню')  
kb\_client = ReplyKeyboardMarkup(*resize\_keyboard*=True, *one\_time\_keyboard*=True)  
kb\_client.add(b1).add(b2)  
schedule = AsyncIOScheduler(*timezone*='Europe/Moscow')  
  
  
class TestStates(Helper):  
 mode = HelperMode.snake\_case  
  
 TEST\_STATE\_0 = ListItem()  
 TEST\_STATE\_1 = ListItem()  
 TEST\_STATE\_2 = ListItem()  
 TEST\_STATE\_3 = ListItem()  
 TEST\_STATE\_4 = ListItem()  
 TEST\_STATE\_5 = ListItem()  
  
  
class FSMAdmin(StatesGroup):  
 price = State()  
 tipe = State()  
 text = State()  
  
  
*# start*async def on\_startup():  
 schedule.start()  
 *print*('стартануло')  
  
*# buy*@dp.message\_handler(*commands*=['Оплатить'])  
async def buy(*message*: types.Message,*state*:FSMContext):  
 *print*(TestStates.all())  
 if PAYMENTS\_TOKEN.split(':')[1] == 'TEST':  
 read = await database\_conn.sql\_select\_command4(*state*)  
 ret = read  
 rit = *list*(ret)  
 rat = ','.join(''.join(tup) for tup in rit)  
 await *message*.answer(*text*='Выберите тип игры, который хотите оплатить:'+rat)  
 await FSMAdmin.tipe.set()  
 *#for ret in read:  
 # a = list(ret)  
 # for i in range(len(a)):  
 # a[i] = str(a[i])  
 #for x in ret:  
 #print(x)  
 #await bot.send\_message(message.chat.id, x)  
 #print(ret)  
 #print(rit)  
  
 print*(rat)  
 *#print(rut)* await *message*.answer('Введи свое сообщение')  
 await take\_message(*message*,*state*)  
  
  
  
  
async def take\_message(*message*: types.Message, *state*: FSMContext):  
 *# Устанавливаем состояние ожидания* async with *state*.proxy() as data:  
 data['text'] = *message*.text  
  
 *# Выключаем состояние  
 print*(data)  
  
 *# b = x['tipe']['text']  
 # c = str("".join(x['tipe']['text']))  
 # text = await message  
 # edited\_type = x.replace(x, text)  
 # b['text']= await FSMAdmin.tipe.set()  
 # x['gtype'] = x['tipe'] + b['text']  
  
 # print(edited\_type)  
  
 # print(c)* async with *state*.proxy() as x2:  
 await FSMAdmin.tipe.set()  
 *# await bot.send\_message(message.chat.id, "Тестовый платеж")  
 # await database\_conn.sql\_select\_command2(x['tipe'])  
 # print(x['price'])* await *state*.finish()  
 '''await bot.send\_invoice(message.chat.id, title="Участие в одной игре",  
 description="Оплата на участие в одной игре",  
 provider\_token=PAYMENTS\_TOKEN, currency="rub",  
 is\_flexible=False, prices=x['price'] \* 100,  
 start\_parameter="one-month-subscription",  
 payload="test-invoice-payload") #keyError 'price'  
 '''  
  
  
*# pre checkout*@dp.pre\_checkout\_query\_handler(lambda *query*: True)  
async def pre\_checkout\_query(*pre\_checkout\_q*: types.PreCheckoutQuery):  
 await bot.answer\_pre\_checkout\_query(*pre\_checkout\_q*.id, *ok*=True)  
  
 *print*('хуйня')  
 *# successful payment*@dp.message\_handler(*content\_types*=ContentType.SUCCESSFUL\_PAYMENT)  
async def successful\_payment(*message*: types.Message):  
 *print*("SUCCESSFUL PAYMENT")  
 payment\_info = *message*.successful\_payment.to\_python()  
 for k, v in payment\_info.items():  
 *print*(f"{k}= {v}")  
 await bot.send\_message(*message*.chat.id,  
 f"Платеж на сумму {*message*.successful\_payment.total\_amount // 100}{*message*.successful\_payment.currency} провел успешно ")  
 if True:  
 schedule.add\_job(apsched.timer\_message\_cron, *trigger*='cron', *seconds*=10, *kwargs*={'message': *message*})  
 *print*('запустилось')  
 schedule.start()  
 *print*('стартануло, но 2')  
 await asyncio.sleep(1)  
 if True:  
 schedule.add\_job(rules.rules, *trigger*='interval', *seconds*=10, *kwargs*={'bot': bot})  
 *print*('запустилось, но 2')  
 await asyncio.sleep(1)  
 if True:  
 schedule.add\_job(kick\_user.kick\_user, *trigger*='interval', *seconds*=10, *kwargs*={'bot': bot})  
 *print*('запустилось, но 3')  
 await asyncio.sleep(1)  
  
  
  
  
*# print('ошибка в ссылке')  
# link = await bot.get\_chat(chat\_id=message.from\_user.id)  
# print(link)  
# await message.reply('ошибка: некорректная ссылка')  
# timestamp polling*"""@dp.message\_handler(commands=["start"])  
async def start\_handler(message: types.Message):  
 user\_id = message.from\_user.id  
 user\_name = message.from\_user.first\_name  
 for i in range(7):  
  
 await asyncio.sleep(60 \* 1 \* 1)  
 await bot.send\_message(user\_id, 'игра началась')  
 # executor.start\_polling(dp, skip\_updates=True)"""  
  
  
def register\_handlers\_ckb(*dp*: Dispatcher):  
 *# dp.register\_message\_handler(answer\_to\_answers, state=FSMAdmin.text)  
 """dp.register\_message\_handler(url\_command)"""* if \_\_name\_\_ == '\_\_main\_\_':  
 executor.start\_polling(*dp*, *skip\_updates*=True, *on\_startup*=on\_startup)