**Исполнитель:**  
студент\_ка группы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ /  
«\_\_» \_\_\_\_\_\_\_\_\_2034 г.

|  |  |
| --- | --- |
| *Подп. и дата* |  |
| *Инв. № дубл.* |  |
| *Взам. инв. №* |  |
| *Подп. и дата* |  |
| *Инв. № подл* |  |

**Москва 2034**

СОГЛАСОВАНО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
«\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2034 г.

УТВЕРЖДАЮ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
«\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2034 г.

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ import telebot  
from telebot import types  
from docxtpl import DocxTemplate  
from dotenv import load\_dotenv  
import os  
import logging  
import time  
from tech\_spec\_data import tech\_spec\_questions  
from explanatory\_note\_data import explanatory\_note\_questions  
from title\_page\_data import title\_page\_questions  
from docx2pdf import convert  
  
OUTPUT\_FOLDER = "output\_files"  
  
if not os.path.exists(OUTPUT\_FOLDER):  
 os.makedirs(OUTPUT\_FOLDER)  
  
logging.basicConfig(filename='errors.log', level=logging.ERROR, format='%(asctime)s - %(message)s',  
 datefmt='%Y-%m-%d %H:%M:%S')  
  
load\_dotenv()  
bot = telebot.TeleBot(os.getenv('TOKEN'))  
last\_chat\_id = None  
  
chat\_steps = {}  
  
current\_document\_type = {}  
  
title\_page\_code\_blocks = {}  
tech\_spec\_code\_blocks = {}  
explanatory\_note\_code\_blocks = {}  
  
  
def create\_code\_blocks(chat\_id):  
 title\_page\_code\_blocks[chat\_id] = {key: '' for \_, key in title\_page\_questions}  
 tech\_spec\_code\_blocks[chat\_id] = {key: '' for \_, key in tech\_spec\_questions}  
 explanatory\_note\_code\_blocks[chat\_id] = {key: '' for \_, key in explanatory\_note\_questions}  
  
  
def create\_ask\_and\_save\_handlers(code\_blocks, question, code\_block\_key):  
 def ask\_handler(message):  
 msg = bot.send\_message(message.chat.id, question, parse\_mode='HTML')  
 bot.register\_next\_step\_handler(msg, save\_handler)  
  
 def save\_handler(message):  
 if message.text.startswith("/start"):  
 handle\_start\_after\_creation(message)  
 return  
 if message.text.startswith("/stop"):  
 handle\_stop(message)  
 return  
 chat\_id = message.chat.id  
 if chat\_id not in code\_blocks:  
 code\_blocks[chat\_id] = {}  
 code\_blocks[chat\_id][code\_block\_key] = message.text  
 next\_step\_handler(message, chat\_steps[message.chat.id])  
  
 return ask\_handler, save\_handler  
  
  
title\_page\_handlers = [create\_ask\_and\_save\_handlers(title\_page\_code\_blocks, question, key) for question, key  
 in title\_page\_questions]  
tech\_spec\_handlers = [create\_ask\_and\_save\_handlers(tech\_spec\_code\_blocks, question, key) for question, key in  
 tech\_spec\_questions]  
explanatory\_note\_handlers = [create\_ask\_and\_save\_handlers(explanatory\_note\_code\_blocks, question, key) for question, key  
 in explanatory\_note\_questions]  
  
  
def next\_step\_handler(message, step=0):  
 global last\_chat\_id  
 last\_chat\_id = message.chat.id  
 chat\_steps[message.chat.id] = step + 1  
 document\_type = current\_document\_type[message.chat.id]  
 if document\_type == "title\_page":  
 if step len(title\_page\_handlers):  
 ask\_handler, \_ = title\_page\_handlers[step]  
 ask\_handler(message)  
 elif step == len(title\_page\_handlers):  
 create\_document(message, title\_page\_code\_blocks[message.chat.id], 'title\_page\_maket.docx',  
 'title\_page')  
 elif document\_type == "technical\_specifications":  
 if step len(tech\_spec\_handlers):  
 ask\_handler, \_ = tech\_spec\_handlers[step]  
 ask\_handler(message)  
 elif step == len(tech\_spec\_handlers):  
 create\_document(message, tech\_spec\_code\_blocks[message.chat.id], 'tech\_spec\_maket.docx',  
 'technical\_specifications')  
 elif document\_type == "explanatory\_note":  
 if step len(explanatory\_note\_handlers):  
 ask\_handler, \_ = explanatory\_note\_handlers[step]  
 ask\_handler(message)  
 elif step == len(explanatory\_note\_handlers):  
 create\_document(message, explanatory\_note\_code\_blocks[message.chat.id], 'explanatory\_note\_maket.docx',  
 'explanatory\_note')  
  
  
def start(message):  
 keyboard = types.InlineKeyboardMarkup()  
 button1 = types.InlineKeyboardButton("Титульный лист", callback\_data="title\_page")  
 button2 = types.InlineKeyboardButton("Техническое задание", callback\_data="technical\_specifications")  
 button3 = types.InlineKeyboardButton("Пояснительная записка", callback\_data="explanatory\_note")  
  
 keyboard.add(button1)  
 keyboard.add(**  
message(message.chat.id, 'Процесс создания документа прерван.')  
 document\_type = current\_document\_type[message.chat.id]  
 if document\_type == "technical\_specifications":  
 create\_document(message, tech\_spec\_code\_blocks[message.chat.id], 'tech\_spec\_maket.docx',  
 'technical\_specifications')  
 elif document\_type == "explanatory\_note":  
 create\_document(message, explanatory\_note\_code\_blocks[message.chat.id], 'explanatory\_note\_maket.docx',  
 'explanatory\_note')  
 elif document\_type == "title\_page":  
 create\_document(message, title\_page\_code\_blocks[message.chat.id], 'title\_page\_maket.docx',  
 'title\_page')  
  
  
@bot.callback\_query\_handler(func=lambda call: True)  
def document\_template\_handler(call):  
 global last\_chat\_id  
 last\_chat\_id = call.message.chat.id  
 document\_type = call.data  
  
 if document\_type == "ignore":  
 return # Завершить обработчик, если callback\_data равно "ignore"  
  
 update\_document\_choice\_to\_inactive(call.message.chat.id, call.message.message\_id, document\_type)  
  
 current\_document\_type[call.message.chat.id] = document\_type  
 if document\_type == "technical\_specifications":  
 bot.send\_message(call.message.chat.id,  
 "Приступим к созданию вашего технического задания.\n"  
 "Сейчас мы поэтапно пройдём по всем пунктам, начиная с титульного листа и заканчивая списком литературы. "  
 "Следуйте дальнейшим указаниям и подсказкам. \n\nВ любом месте вы можете остановиться и отправить ответ позднее, "  
 "а если захотите закончить и получить не до конца заполненный документ, введите /stop. \n\n"  
 "Если вы хотите пропустить любой шаг, необходимо отправить в ответ любой символ, тогда вы сможете "  
 "заполнить этот раздел самостоятельно в итоговом документе.",  
 parse\_mode='HTML')  
 chat\_steps[call.message.chat.id] = 0  
 next\_step\_handler(call.message, step=0)  
 elif document\_type == "explanatory\_note":  
 bot.send\_message(call.message.chat.id,  
 "Приступим к созданию вашей пояснительной записки.\n Сейчас мы поэтапно пройдём по всем пунктам, "  
 "начиная с титульного листа и заканчивая списком литературы. Следуйте дальнейшим указаниям и подсказкам. "  
 "\n\nВ любом месте вы можете остановиться и отправить ответ позднее, а если захотите закончить и получить "  
 "не до конца заполненный документ, введите /stop. \n\nЕсли вы хотите пропустить любой шаг, необходимо отправить "  
 "в ответ любой символ, тогда вы сможете заполнить этот раздел самостоятельно в итоговом документе.",  
 parse\_mode='HTML')  
 chat\_steps[call.message.chat.id] = 0  
 next\_step\_handler(call.message, step=0)  
 elif document\_type == "title\_page":  
 bot.send\_message(call.message.chat.id,  
 "Приступим к созданию вашего титульного листа.\nСейчас мы поэтапно пройдём по всем пунктам, "  
 "следуйте дальнейшим указаниям и подсказкам. \n\nВ любом месте вы можете остановиться и отправить ответ "  
 "позднее, а если захотите закончить и получить не до конца заполненный документ, введите /stop. \n\nЕсли вы "  
 "хотите пропустить любой шаг, необходимо отправить в ответ любой символ, тогда вы сможете заполнить этот раздел "  
 "самостоятельно в итоговом документе.",  
 parse\_mode='HTML')  
 chat\_steps[call.message.chat.id] = 0  
 next\_step\_handler(call.message, step=0)  
  
  
@bot.message\_handler(commands=['start'])  
def handle\_start(message):  
 global last\_chat\_id  
 last\_chat\_id = message.chat.id  
 create\_code\_blocks(message.chat.id)  
 start(message)  
  
  
@bot.message\_handler(commands=['stop'])  
def handle\_stop\_command(message):  
 handle\_stop(message)  
  
  
@bot.me