Код Ревью TestBot

1. Структура проекта
    1. Должна быть основная папка, обычно называю src, все питон файлы там
    2. Пример

```
.
├── docker-compose.yml
├── Dockerfile
├── .env
├── .env.dist
├── .gitignore
├── requirements.txt
├── README.md
└── src
    ├── config.py
    ├── database/
    ├── filters/
    ├── handlers/
    ├── __main__.py
    ├── middlewares/
    └── states/
```

    В этом случае все по папкам раскидывать не надо, но структуры такой придерживаться

    3. Нет файла **requirements.txt**
2. Используй coventional commits при названии коммитов (ссылка)
3. Нет .env файла, где хранятся все чувствительные данные. Токен, айди админов
4. В проекте упоминается файл config.py, но в репозитории его нет
5. Если нужно объяснить что делает клавиатура
   Так не делать:

```python
def get_phone_keyboard():
    # Клавиатура для запроса номера телефона
    return ReplyKeyboardMarkup(keyboard=[[...]])
```

   Так делать:

```python
def get_phone_keyboard():
    """Клавиатура для запроса номера телефона"""
    return ReplyKeyboardMarkup(keyboard=[[...]])
```

6. Все импорты в начале файла, точно не по центру функции

```
236         from keyboards import get_options_inline_keyboard
237         await message.answer(
238             "Выберите один или несколько пунктов (можно нажи
239             reply_markup=get_options_inline_keyboard()
240         )
241         await state.set_state(RequestForm.choosing_options)
```

7. Также к вопросу об .env и конфигу

```
    await callback.answer()
    return
ADMIN_CHAT_ID = -1002755127121
rus_options = [option_map.get(opt, opt)
text = f"Заявка №{request_id}\nОт: {ful
```

8. Файл main.py слишком большой, нужно его разделить на несколько
9. Серьёзная ошибка, при каждом запросе ты подключаешься к БД, это ресурсоёмкий процесс, который занимает время. Бот подключается один раз к БД и отключается при выключении

```
async def register_user(user_id: int, full_name: str, birth_date: str, phone
    conn = await asyncpg.connect(**DB_CONFIG)
    try:
        await conn.execute('''
            INSERT INTO users (user_id, full_name, birth_date, phone_number)
            VALUES ($1, $2, $3, $4)
        ''', user_id, full_name, birth_date, phone_number)
    finally:
        await conn.close()

async def save_request(user_id: int, request_type: str, screenshot_file_id:
    conn = await asyncpg.connect(**DB_CONFIG)
    try:
        row = await conn.fetchrow('''
            INSERT INTO requests (user_id, request_type, screenshot_file_id,
            VALUES ($1, $2, $3, $4)
            RETURNING id
        ''', user_id, request_type, screenshot_file_id, ", ".join(options))
        return row['id']
    finally:
        await conn.close()
```

10. Небольшие улучшения внешнего вида кода:
    - Последняя строка в питон файлах должна быть пустой

- Сортировка импортов, сначала импорт библиотек, а потом самописных модулей, пример

```
import re
from datetime import datetime

from aiogram import F, Router
from aiogram.fsm.context import FSMContext
from aiogram.types import InlineKeyboardMarkup, Inline

from src.config import AUTOPOSTING_MENU_MESSAGE
from src.database.userDB import UserDB
from src.handlers.user_handlers import delete_prev_key
from src.keyboards.kb import get_autopost_menu, get_au
from src.states.states import ScheduleStates
```

11. Улучшение логики
    - Много хендлеров которые отвечают за несколько действий, старайся делать так, чтобы одна функция отвечала за определенное конкретное действие
    - Используй вместо lambda функции магический фильтр F

```
@dp.callback_query(lambda c: c.data.startswith("admin_"))
async def admin_callback_handler(callback: CallbackQuery,
    if not await is_admin(callback.from_user.id):
        await callback.answer("❌ У вас нет доступа к адми
        return
```

```
@user_styles_router.callback_query(F.data.startswith("styles_page_"))
async def show_styles_page(callback: CallbackQuery, state: FSMContext,
    await state.clear()
```