Import logging

From telegram import Update, ReplyKeyboardMarkup, InlineKeyboardButton, InlineKeyboardMarkup

From telegram.ext import Application, CommandHandler, MessageHandler, filters, ContextTypes, CallbackQueryHandler

From config import BOT\_TOKEN, ADMIN\_ID, PAYMENT\_LINK

Import database

Import sqlite3

Import datetime

# Настройка логгирования

Logging.basicConfig(format=»%(asctime)s - %(name)s - %(levelname)s - %(message)s», level=logging.INFO)

Logger = logging.getLogger(\_\_name\_\_)

# Состояния бота

VACANCY, SAILOR\_PROFILE = range(2)

# Инициализация базы данных

Database.init\_db()

# Команда /start

Async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE):

Keyboard = [

[«🏢 Я компания», «👤 Я моряк»],

[«🔍 Поиск вакансий», «👀 Посмотреть резюме»]

]

Await update.message.reply\_text(

«👋 Добро пожаловать в SeaJobsBot!\nВыберите действие:»,

Reply\_markup=ReplyKeyboardMarkup(keyboard, resize\_keyboard=True, one\_time\_keyboard=False)

)

# Обработка роли «Я компания»

Async def handle\_company(update: Update, context: ContextTypes.DEFAULT\_TYPE):

Await update.message.reply\_text(

«📝 Введите данные вакансии в формате:\n\n»

«Название компании:\n»

«Контакт для связи:\n»

«Должность:\n»

«Тип судна:\n»

«Район плавания:\n»

«Зарплата (USD):\n»

«Требуемый опыт (лет):\n\n»

«Пример:\n»

«Морская Компания XYZ\n»

«@hr\_manager\n»

«Старший механик\n»

«Танкер\n»

«Средиземное море\n»

«5000\n»

«5»

)

Return VACANCY

# Сохранение вакансии

Async def save\_vacancy(update: Update, context: ContextTypes.DEFAULT\_TYPE):

Try:

Data = update.message.text.split('\n')

If len(data) < 7:

Await update.message.reply\_text(«❌ Недостаточно данных. Пожалуйста, введите все 7 пунктов»)

Return VACANCY

Conn = sqlite3.connect('seafarers.db')

C = conn.cursor()

c.execute(«INSERT INTO vacancies (user\_id, company\_name, contact, position, vessel\_type, sailing\_area, salary, experience\_required) VALUES (?, ?, ?, ?, ?, ?, ?, ?)»,

(update.effective\_user.id, data[0], data[1], data[2], data[3], data[4], int(data[5]), int(data[6])))

Conn.commit()

Await update.message.reply\_text(«✅ Вакансия сохранена! Она будет доступна морякам после модерации.»)

# Уведомление админу

Await context.bot.send\_message(

ADMIN\_ID,

F»🔥 Новая вакансия!\nКомпания: {data[0]}\nДолжность: {data[2]}\nЗарплата: {data[5]} USD»

)

Except Exception as e:

Await update.message.reply\_text(f»❌ Ошибка: {e}»)

Return ConversationHandler.END

# Обработка роли «Я моряк»

Async def handle\_sailor(update: Update, context: ContextTypes.DEFAULT\_TYPE):

Await update.message.reply\_text(

«📝 Введите ваше резюме в формате:\n\n»

«ФИО:\n»

«Контакт для связи:\n»

«Желаемая должность:\n»

«Опыт работы (лет):\n»

«Желаемая зарплата (USD):\n\n»

«Пример:\n»

«Иванов Иван Иванович\n»

«@ivanov\n»

«Старший механик\n»

«7\n»

«5000»

)

Return SAILOR\_PROFILE

# Сохранение резюме моряка

Async def save\_sailor(update: Update, context: ContextTypes.DEFAULT\_TYPE):

Try:

Data = update.message.text.split('\n')

If len(data) < 5:

Await update.message.reply\_text(«❌ Недостаточно данных. Пожалуйста, введите все 5 пунктов»)

Return SAILOR\_PROFILE

Conn = sqlite3.connect('seafarers.db')

C = conn.cursor()

c.execute(«INSERT INTO sailors (user\_id, full\_name, contact, position, experience, desired\_salary) VALUES (?, ?, ?, ?, ?, ?)»,

(update.effective\_user.id, data[0], data[1], data[2], int(data[3]), int(data[4])))

Conn.commit()

Await update.message.reply\_text(«✅ Ваше резюме сохранено! Оно будет доступно компаниям.»)

# Уведомление админу

Await context.bot.send\_message(

ADMIN\_ID,

F»👤 Новый моряк!\nФИО: {data[0]}\nДолжность: {data[2]}\nОпыт: {data[3]} лет»

)

Except Exception as e:

Await update.message.reply\_text(f»❌ Ошибка: {e}»)

Return ConversationHandler.END

# Поиск вакансий

Async def search\_vacancies(update: Update, context: ContextTypes.DEFAULT\_TYPE):

Conn = sqlite3.connect('seafarers.db')

C = conn.cursor()

c.execute(«SELECT \* FROM vacancies ORDER BY created\_at DESC LIMIT 5»)

vacancies = c.fetchall()

if not vacancies:

await update.message.reply\_text(«😔 Пока нет доступных вакансий»)

return

response = «🔍 Последние вакансии:\n\n»

for vac in vacancies:

response += f»🏢 {vac[2]}\n⚓ {vac[4]}\n🌍 {vac[6]}\n💵 {vac[7]} USD\n\n»

response += f»Для просмотра контактов: /contact\_{vac[0]}\n»

response += «―»\*20 + «\n»

await update.message.reply\_text(response)

# Обработка кнопки оплаты

Async def handle\_payment(update: Update, context: ContextTypes.DEFAULT\_TYPE):

Query = update.callback\_query

Await query.answer()

Vacancy\_id = query.data.split(«\_»)[1]

Await query.edit\_message\_text(

F»💳 Для просмотра контактов оплатите 300 руб:\n{PAYMENT\_LINK}\n\n»

«После оплаты отправьте скриншот сюда для подтверждения.»,

Reply\_markup=InlineKeyboardMarkup([

[InlineKeyboardButton(«✅ Я оплатил», callback\_data=f»confirmed\_{vacancy\_id}»)]

])

)

# Подтверждение оплаты

Async def confirm\_payment(update: Update, context: ContextTypes.DEFAULT\_TYPE):

Query = update.callback\_query

Await query.answer()

Vacancy\_id = query.data.split(«\_»)[1]

Conn = sqlite3.connect('seafarers.db')

C = conn.cursor()

# Получаем контакты компании

c.execute(«SELECT contact FROM vacancies WHERE id=?», (vacancy\_id,))

contact = c.fetchone()[0]

# Сохраняем платеж

c.execute(«INSERT INTO payments (user\_id, amount, vacancy\_id) VALUES (?, ?, ?)»,

(query.from\_user.id, 300, vacancy\_id))

Conn.commit()

Await query.edit\_message\_text(

F»🔓 Контакты компании:\n{contact}\n\n»

«Сообщите при обращении, что нашли их через SeaJobsBot!»

)

# Уведомление админу

Await context.bot.send\_message(

ADMIN\_ID,

F»💰 Новый платеж!\nПользователь: @{query.from\_user.username}\nВакансия ID: {vacancy\_id}»

)

Def main():

Application = Application.builder().token(BOT\_TOKEN).build()

# Обработчики команд

Application.add\_handler(CommandHandler(«start», start))

# Обработчики сообщений

Application.add\_handler(MessageHandler(filters.Regex(«^🏢 Я компания$»), handle\_company))

Application.add\_handler(MessageHandler(filters.Regex(«^👤 Я моряк$»), handle\_sailor))

Application.add\_handler(MessageHandler(filters.Regex(«^🔍 Поиск вакансий$»), search\_vacancies))

# Обработчики состояний

Conv\_handler = ConversationHandler(

Entry\_points=[],

States={

VACANCY: [MessageHandler(filters.TEXT & ~filters.COMMAND, save\_vacancy)],

SAILOR\_PROFILE: [MessageHandler(filters.TEXT & ~filters.COMMAND, save\_sailor)]

},

Fallbacks=[]

)

Application.add\_handler(conv\_handler)

# Обработчики колбэков

Application.add\_handler(CallbackQueryHandler(handle\_payment, pattern=»^pay\_»))

Application.add\_handler(CallbackQueryHandler(confirm\_payment, pattern=»^confirmed\_»))

# Запуск бота

Application.run\_polling()

If \_\_name\_\_ == «\_\_main\_\_»:

Main()