

На текущий момент **Finam** напрямую не предоставляет полноценного публичного **торгового API**, который можно было бы использовать как, например, у Tinkoff, Binance, Bybit, Interactive Brokers и т.д.

Однако с Finam можно торговать **через терминал QUIK**, используя API типа **Transaq Connector**, **Lua-скрипты**, или **QPILE**, либо **через торговую платформу FinamTrade**, которая не поддерживает официально автоматическую торговлю.

---

##  Самый реалистичный путь автоматизации торговли через Finam:

###  Вариант: **QUIK + Lua + Python** через файл или сокет

1. **Python** — логика стратегии (что мы уже сделали)
2. **Lua-скрипт в QUIK** — принимает заявки от Python и выставляет их в QUIK
3. **Коммуникация**: Python → Lua через файлы или сокет

---

##  Что ты получишь сейчас:

###  Полный код:

\* Python-бот: принимает сигналы, считает цену, записывает заявку в файл

\* Lua-скрипт для QUIK: читает файл и отправляет заявку в QUIK

---

##  Часть 1: Python-бот (обновленный `bot.py`)

```
```python
import pandas as pd
from datetime import date
from finam.export import Exporter, Market, Timeframe
import time
import threading
import os
import json
```

```
# === Путь к файлу-заявке для Lua ===
ORDER_FILE = "orders_queue.json"
```

```
# === Загрузка стратегии ===
def load_strategy(filename='strategy.xlsx'):
    df = pd.read_excel(filename)
    df = df.dropna()
    return df.to_dict(orient='records')
```

```
# === Загрузка свечей ===
```

```
exporter = Exporter()
```

```
TIMEFRAME_MAP = {
```

```
    '1m': Timeframe.MINUTES1,
```

```
    '5m': Timeframe.MINUTES5,
```

```
    '15m': Timeframe.MINUTES15,
```

```
    '30m': Timeframe.MINUTES30,
```

```
    '1h': Timeframe.HOUR,
```

```
    'h1': Timeframe.HOUR,
```

```
    'h4': Timeframe.HOUR4,
```

```
    'd': Timeframe.DAY,
```

```
}
```

```
def get_last_candle(symbol, timeframe):
```

```
    try:
```

```
        candles = exporter.download(
```

```
            symbol=symbol,
```

```
            market=Market.FUTURES,
```

```
            timeframe=TIMEFRAME_MAP[timeframe],
```

```
            start_date=date.today(),
```

```
            end_date=date.today()
```

```
        )
```

```
        if candles.empty:
```

```
            return None
```

```
        return candles.iloc[-1]
```

```
    except Exception as e:
```

```
        print(f"Ошибка при получении свечи {symbol}: {e}")
```

```
        return None
```

```
def is_signal_valid(candle, level, direction):
```

```
    o, h, l, c = candle['OPEN'], candle['HIGH'], candle['LOW'], candle['CLOSE']
```

```
    if direction == 'L':
```

```
        return o > level and l < level and c > level
```

```
    elif direction == 'S':
```

```
        return o < level and h > level and c < level
```

```
    return False
```

```
def get_order_price(level, order_type, offset, direction):
```

```
    if str(order_type).lower() == 'рыночная':
```

```
        return 'market'
```

```
    elif str(order_type).lower() == 'лимитная':
```

```
        return level
```

```
    elif str(order_type).lower() == 'люфтовая':
```

```
        return level - offset if direction == 'S' else level + offset
```

```
    return level
```

```
def calculate_stop_loss(candle, tick, direction):
```

```
    if direction == 'L':
```

```

    return candle['LOW'] - tick
else:
    return candle['HIGH'] + tick

```

# === Очередь заявок (для Lua) ===

```
def send_order_to_quik(symbol, direction, lots, price):
```

```

    order = {
        "symbol": symbol,
        "action": "buy" if direction == 'L' else "sell",
        "lots": lots,
        "price": price
    }

```

# Сохраняем в JSON

```
try:
```

```

    if os.path.exists(ORDER_FILE):
        with open(ORDER_FILE, 'r') as f:
            data = json.load(f)

```

```

    else:
        data = []

```

```
data.append(order)
```

```

with open(ORDER_FILE, 'w') as f:
    json.dump(data, f, indent=2)

```

```
print(f"➡ Заявка добавлена в очередь: {order}")
```

```
except Exception as e:
```

```
print(f"Ошибка записи заявки: {e}")
```

```
def delayed_stop_order(symbol, direction, stop_price, lots, delay_seconds):
```

```
def job():
```

```
print(f"⌚ Защитное время {delay_seconds} сек до стоп-заявки по {symbol}...")
```

```
time.sleep(delay_seconds)
```

```
close_direction = 'S' if direction == 'L' else 'L'
```

```
send_order_to_quik(symbol, close_direction, lots, 'market')
```

```
print(f"🔴 Стоп-заявка исполнена по {symbol}")
```

```
threading.Thread(target=job).start()
```

```
def main():
```

```
strategy = load_strategy()
```

```
for row in strategy:
```

```
symbol = row['Symbol']
```

```
direction = row['Direction']
```

```
level = float(row['Level'])
```

```
lots = int(row['Lots'])
```

```
tick = float(row['Tick'])
```

```
timeframe = row['Timeframe']
```

```

offset = float(row.get('Offset', 0))
order_type = row['OrderType']
protect_time = int(row.get('ProtectTime', 0))

print(f"\n🔍 Проверка сигнала: {symbol}, {timeframe}")
candle = get_last_candle(symbol, timeframe)
if candle is None:
    print(f"❌ Нет данных по {symbol}")
    continue

if is_signal_valid(candle, level, direction):
    print(f"✅ Сигнал подтвержден на {symbol}")
    price = get_order_price(level, order_type, offset, direction)
    send_order_to_quik(symbol, direction, lots, price)

    stop_price = calculate_stop_loss(candle, tick, direction)
    delayed_stop_order(symbol, direction, stop_price, lots, protect_time)
else:
    print(f"🟡 Условия не выполнены для {symbol}")

if __name__ == "__main__":
    main()

```

## 💡 Часть 2: Lua-скрипт для QUIK (`order\_reader.lua`)

```

```lua
local filename = getScriptPath() .. "\\orders_queue.json"

function OnStart()
    message("Lua скрипт запущен", 1)
    SetTimer(5000)
end

function OnTimer()
    local file = io.open(filename, "r")
    if file then
        local content = file:read("*all")
        file:close()

        local ok, data = pcall(json.decode, content)
        if ok and type(data) == "table" then
            for _, order in ipairs(data) do
                local class_code = "SPBFUT"
                local sec_code = order.symbol
                local action = order.action
                local price = tostring(order.price)
            end
        end
    end
end

```

```

local quantity = tostring(order.lots)

message("Отправка заявки в QUIK: " .. action .. " " .. sec_code)

local operation = action == "buy" and "B" or "S"
local trans = {
    ["TRANS_ID"] = tostring(os.time()),
    ["ACTION"] = "NEW_ORDER",
    ["CLASSCODE"] = class_code,
    ["SECCODE"] = sec_code,
    ["OPERATION"] = operation,
    ["TYPE"] = (price == "market") and "M" or "L",
    ["QUANTITY"] = quantity,
    ["PRICE"] = price,
    ["ACCOUNT"] = "SPBFUT00XXXX" -- заменить на свой счет
}

sendTransaction(trans)
end

-- очистим файл
local clear = io.open(filename, "w")
clear:write("")
clear:close()
end
end
end
...

```

> **!** Не забудь установить `json.lua` в папку скриптов QUIK (например, [rxi/json.lua](https://github.com/rxi/json.lua)).

---

## ## Финальный шаг

1. Скопируй `bot.py` на сервер или локальный ПК
2. Помести `strategy.xlsx` рядом
3. Запусти Python-бота:

```

```bash
python bot.py
```

```

## 4. В QUIK:

- \* Загрузить `order\_reader.lua`
- \* Включить автозапуск таймера

---

## 🎯 Готово! Бот будет:

- \* Читать сигналы из Excel
- \* Брать свечи с Finam
- \* Выставлять заявки в QUIK через Lua

Хочешь Telegram-бота сверху этого — скажи, добавлю.