```
import requests
import time
import logging
import os
from datetime import datetime
import pytz

API_KEY = os.environ.get('TRADING212_API_KEY', '')
BASE_URL = 'https://live.trading212.com/api/v0'
SYMBOL = 'RGLD'
TAKE_PROFIT = 1.0
STOP_LOSS = 1.0
CHECK_INTERVAL = 60

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(message)s')

headers = {
    'Authorization': API_KEY,
    'Content-Type': 'application/json'
}

def is_market_open():
    rome = pytz.timezone('Europe/Rome')
    now = datetime.now(rome)
    if now.weekday() >= 5:
        return False
    if now.hour < 15 or now.hour >= 22:
        return False
    if now.hour == 15 and now.minute < 30:
        return False
    return True

def get_account_cash():
    r = requests.get(f'{BASE_URL}/equity/account/cash', headers=headers)
    logging.info(f'Cash status: {r.status_code} - {r.text}')
    return r.json()

def get_positions():
    r = requests.get(f'{BASE_URL}/equity/portfolio', headers=headers)
    logging.info(f'Portfolio status: {r.status_code} - {r.text}')
    return r.json()

def place_buy_order(quantity):
    payload = {
```

```
        'ticker': SYMBOL,
        'quantity': quantity,
        'timeValidity': 'DAY'
    }
    r = requests.post(f'{BASE_URL}/equity/orders/market', headers=headers, json=p
    logging.info(f'Buy order: {r.text}')
    return r.json()

def place_sell_order(quantity):
    payload = {
        'ticker': SYMBOL,
        'quantity': quantity,
        'timeValidity': 'DAY'
    }
    r = requests.post(f'{BASE_URL}/equity/orders/market', headers=headers, json=p
    logging.info(f'Sell order: {r.text}')
    return r.json()

def run_bot():
    logging.info('Bot avviato - Trading RGLD su Trading 212 Invest')

    while True:
        try:
            if not is_market_open():
                logging.info('Mercato chiuso, attendo...')
                time.sleep(300)
                continue

            cash_data = get_account_cash()
            free_cash = float(cash_data.get('free', 0))
            logging.info(f'Cash disponibile: {free_cash:.2f} EUR')

            positions = get_positions()
            rgld_position = None
            if isinstance(positions, list):
                for pos in positions:
                    if pos.get('ticker') == SYMBOL:
                        rgld_position = pos
                        break

            if rgld_position:
                ppl = float(rgld_position.get('ppl', 0))
                qty = float(rgld_position.get('quantity', 0))
                logging.info(f'Posizione RGLD - P&L: {ppl:.2f} EUR - Qty: {qty}')

                if ppl >= TAKE_PROFIT:
                    logging.info(f'Take profit! +{ppl:.2f} EUR')
```

```
                    place_sell_order(qty)
                elif ppl <= -STOP_LOSS:
                    logging.info(f'Stop loss! {ppl:.2f} EUR')
                    place_sell_order(qty)
                else:
                    logging.info(f'In attesa... P&L: {ppl:.2f} EUR')
            else:
                if free_cash >= 10:
                    trade_amount = min(free_cash * 0.2, 20)
                    logging.info(f'Apro posizione con {trade_amount:.2f} EUR')
                    place_buy_order(0.01)
                else:
                    logging.info(f'Cash insufficiente: {free_cash:.2f} EUR')

        except Exception as e:
            logging.error(f'Errore: {e}')

        time.sleep(CHECK_INTERVAL)

if __name__ == '__main__':
    run_bot()
```