

Во время посещения сайта вы соглашаетесь с использованием файлов [cookie](#)

Хорошо



Михаил Шардин ★

личный блог



18 ноября 2025, 05:02

+ Подписаться

Российская алготорговля - это не рынок, а зоопарк API, а FinLabPy - единственный волк

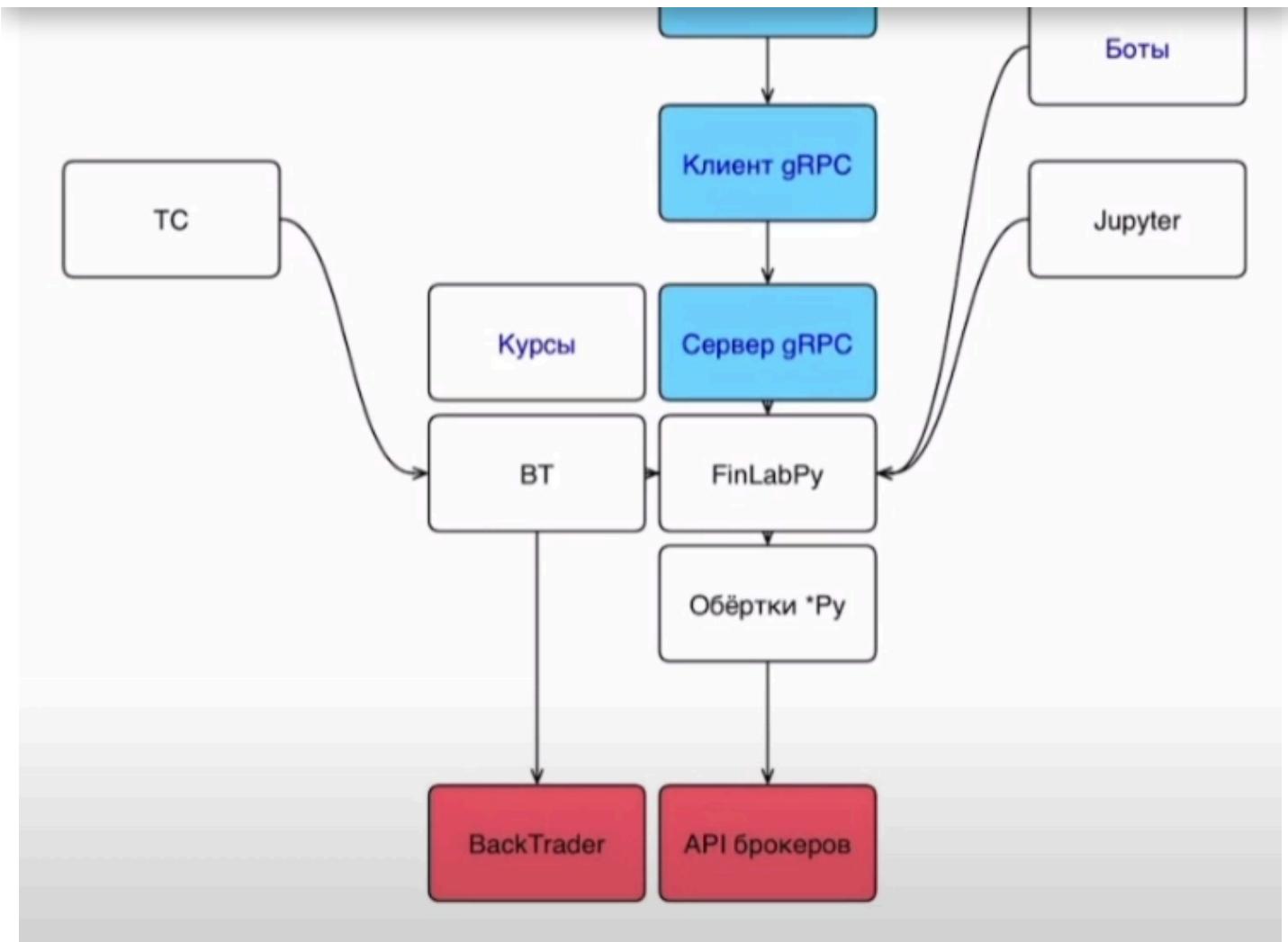
В последнее время я активно занимаюсь автоматизацией торговли и знакомлюсь с разными решениями, два раза летал на конференции, познакомился с интересными людьми. На этом фоне я наткнулся на open-source проект [cia76/FinLabPy](#), о котором уже давно слышал, но никогда не разбирался подробно.

Российская алготорговля переживает странный период: возможности растут, но стандартизации как будто не существует. Брокеры выпускают свои API, но каждый из них живёт в отдельной вселенной – со своим обозначением тикеров, задержками и внезапными отключениями.

Про проблемы алготорговли на Московской бирже почти не пишут, хотя есть мнение что 60% оборота биржи создаётся роботами. А вот [автор этого проекта Игорь Чечет](#) на своём вебинаре рассказывает о том с какими проблемами может столкнуться частный инвестор, когда приходит в алгоритмическую среду.

Начну с главного – какую вообще проблему решает FinLabPy?

Введите текст комментария



Что такое FinLabPy и какую проблему он решает

[cia76/FinLabPy](#) – это унифицированная платформа для анализа рынков, прототипирования торговых идей, тестирования стратегий и запуска автоторговли через нескольких российских брокеров.

Необходимость создания такой библиотеки возникла потому что российские брокеры реализовали API «каждый в меру своих возможностей». Несколько примеров:

- **Финам**: может самостоятельно отваливать подписки.
- **Т-Инвест**: присыпает сделки пачками и с задержкой; бары иногда запаздывают на 2–3 минуты.
- **Алор**: любит перезагружать сервер прямо во время торгов.

FinLabPy по словам его создателя защищает всю эту боль сею: переподключения, нормализация данных, логирование, кэширование, обработка ошибок, единые тикеры – всё это зашито в open-source библиотеке.

Архитектура: три уровня, которые упрощают жизнь разработчику

FinLabPy устроена на трёх уровнях что меня удивило:

1. Нижний слой: нативные API брокеров.

REST, WebSocket, GRPC – всё, что брокер даёт.

2. Средний слой: Python-обёртки.

Отдельные проекты под каждого брокера:

- AlorPy
- FinamPy
- QuikPy
- TinvestPy (Т-Инвест)
- (в работе) обёртка для БКС

3. Верхний слой: FinLabPy

Единый интерфейс, единая модель данных, единая логика. При этом доступ к «универсальным» функциям конкретного брокера сохраняется:

FinLabPy → provider.provider → уникальные методы обёртки.

Вообще меня порадовало, что внутри нет самодельных велосипедов

Технологический стек FinLabPy

Автор сознательно взял лучшие решения рынка и встроил их в экосистему. Никакого изобретения велосипеда.

Основные элементы:

- Python
- Backtrader – тестирование и автоторговля
- TA-Lib – более 200 индикаторов

• Pandas и NumPy

На своём стриме автор поднимает и другие вопросы, например где запускать торговых роботов?

Только на VPS под Linux (Debian/Ubuntu). Причины:

- стабильный интернет,
- нет NAT, DHCP и прочих «домашних» сюрпризов,
- нет DPI и странных блокировок провайдеров,
- скорость соединения «сервер → брокер» всегда выше.

Стоимость VPS: 150–400 рублей в месяц.

Кэширование, спецификации тикеров и работа с расписаниями

FinLabPy строит целую подсистему работы с данными, которая сильно облегчает жизнь.

Кэширование данных. Чтобы:

- уменьшить количество запросов к API брокера;
- ускорить прогон стратегий;
- обходить лимиты (100 запросов в минуту, данные по 1 дню и т. д.).

Кэш хранится в локальных файлах и пополняется пошагово (инкрементально).

```

10     """Работает с барами"""
11     logger = logging.getLogger('FinLabPy.Core')
12     delimiter = '\\'
13     dt_format = '%Y-%m-%d %H:%M:%S'
14
15     class Storage(ABC):
16         Хранитель бар и спецификации тикеров
17         брокера
18
19         def __init__(self, symbol, timeframe, dt_from=None, dt_to=None):
20             super().__init__()
21             self.datapath = os.path.join(os.path.dirname(os.path.realpath(__file__)), '..', '..', 'Data', source, '') # Путь сохранения файлов
22
23             2 из 20
24             def get_bars(self, symbol, time_frame, dt_from=None, dt_to=None):
25                 filename = f'{self.datapath}{symbol.dataname}.{time_frame}.txt' # Полное имя файла
26                 if not os.path.isfile(filename): # Если файл не существует
27                     self.logger.warning(f'Файл {filename} не найден')
28                     return None # То выходим, дальше не продолжаем
29                 self.logger.debug(f'Получение файла {filename}')
30                 file_bars = pd.read_csv(filename) # Импортируем бары из CSV файла в pandas DataFrame
31                 # Имя файла
32                 # Разделитель значений
33                 # Установка колонок
34                 # Дата/время
35                 # Время окончания сессии
36                 # Индексом будет колонка datetime
37                 # Информация о баре
38                 # Следующий бар
39                 # Количество баров
40                 # Индекс первого бара
41                 # Индекс последнего бара
42                 # Количество баров
43                 bars: List[Bar] = []
44                 for index, row in file_bars.iterrows():
45                     if dt_from is not None and index < dt_from: # Пробегаемся по всем полученным барам
46                         continue # То переходим к следующему блоку, дальше не продолжаем
47                     if dt_to is not None and index > dt_to: # Если задана дата/время окончания выборки, и она меньше даты/времени текущего бара
48                         continue # То переходим к следующему блоку, дальше не продолжаем
49                     bars.append(Bar(symbol.board, symbol.symbol, symbol.dataname, time_frame, index, row['open'], row['high'], row['low'], row['close'], row['volume']))
50                 if len(bars) == 0: # Если бары не получены
51                     self.logger.debug(f'Бары отсутствуют')
52                     return None # То выходим, дальше не продолжаем
53                 if dt_from is not None or dt_to is not None: # Если задан фильтр с ... до ...
54                     str_filter = f'>= {dt_from:{self.dt_format}}' if dt_from is not None else f'<= {dt_to:{self.dt_format}}' if dt_to is not None else f'>= {dt_from:{self.dt_format}} &lt;= {dt_to:{self.dt_format}}'
55                     self.logger.debug(f'Фильтр: {str_filter}')
56
57             2 из 20
58             self.filter(str_filter)
59
60             2 из 20
61             def filter(self, str_filter):
62                 bars = [bar for bar in bars if eval(str_filter)]
63
64             2 из 20
65             def __len__(self):
66                 return len(bars)
67
68             2 из 20
69             def __iter__(self):
70                 return iter(bars)
71
72             2 из 20
73             def __getitem__(self, index):
74                 return bars[index]
75
76             2 из 20
77             def __repr__(self):
78                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
79
80             2 из 20
81             def __eq__(self, other):
82                 if not isinstance(other, FileStorage):
83                     return False
84                 if self.symbol != other.symbol:
85                     return False
86                 if self.timeframe != other.timeframe:
87                     return False
88                 if self.dt_from != other.dt_from:
89                     return False
90                 if self.dt_to != other.dt_to:
91                     return False
92                 return True
93
94             2 из 20
95             def __hash__(self):
96                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
97
98             2 из 20
99             def __ne__(self, other):
100                 return not self.__eq__(other)
101
102             2 из 20
103             def __len__(self):
104                 return len(self.bars)
105
106             2 из 20
107             def __iter__(self):
108                 return iter(self.bars)
109
110             2 из 20
111             def __getitem__(self, index):
112                 return self.bars[index]
113
114             2 из 20
115             def __repr__(self):
116                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
117
118             2 из 20
119             def __eq__(self, other):
120                 if not isinstance(other, FileStorage):
121                     return False
122                 if self.symbol != other.symbol:
123                     return False
124                 if self.timeframe != other.timeframe:
125                     return False
126                 if self.dt_from != other.dt_from:
127                     return False
128                 if self.dt_to != other.dt_to:
129                     return False
130                 return True
131
132             2 из 20
133             def __hash__(self):
134                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
135
136             2 из 20
137             def __ne__(self, other):
138                 return not self.__eq__(other)
139
140             2 из 20
141             def __len__(self):
142                 return len(self.bars)
143
144             2 из 20
145             def __iter__(self):
146                 return iter(self.bars)
147
148             2 из 20
149             def __getitem__(self, index):
150                 return self.bars[index]
151
152             2 из 20
153             def __repr__(self):
154                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
155
156             2 из 20
157             def __eq__(self, other):
158                 if not isinstance(other, FileStorage):
159                     return False
160                 if self.symbol != other.symbol:
161                     return False
162                 if self.timeframe != other.timeframe:
163                     return False
164                 if self.dt_from != other.dt_from:
165                     return False
166                 if self.dt_to != other.dt_to:
167                     return False
168                 return True
169
170             2 из 20
171             def __hash__(self):
172                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
173
174             2 из 20
175             def __ne__(self, other):
176                 return not self.__eq__(other)
177
178             2 из 20
179             def __len__(self):
180                 return len(self.bars)
181
182             2 из 20
183             def __iter__(self):
184                 return iter(self.bars)
185
186             2 из 20
187             def __getitem__(self, index):
188                 return self.bars[index]
189
190             2 из 20
191             def __repr__(self):
192                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
193
194             2 из 20
195             def __eq__(self, other):
196                 if not isinstance(other, FileStorage):
197                     return False
198                 if self.symbol != other.symbol:
199                     return False
200                 if self.timeframe != other.timeframe:
201                     return False
202                 if self.dt_from != other.dt_from:
203                     return False
204                 if self.dt_to != other.dt_to:
205                     return False
206                 return True
207
208             2 из 20
209             def __hash__(self):
210                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
211
212             2 из 20
213             def __ne__(self, other):
214                 return not self.__eq__(other)
215
216             2 из 20
217             def __len__(self):
218                 return len(self.bars)
219
220             2 из 20
221             def __iter__(self):
222                 return iter(self.bars)
223
224             2 из 20
225             def __getitem__(self, index):
226                 return self.bars[index]
227
228             2 из 20
229             def __repr__(self):
230                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
231
232             2 из 20
233             def __eq__(self, other):
234                 if not isinstance(other, FileStorage):
235                     return False
236                 if self.symbol != other.symbol:
237                     return False
238                 if self.timeframe != other.timeframe:
239                     return False
240                 if self.dt_from != other.dt_from:
241                     return False
242                 if self.dt_to != other.dt_to:
243                     return False
244                 return True
245
246             2 из 20
247             def __hash__(self):
248                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
249
250             2 из 20
251             def __ne__(self, other):
252                 return not self.__eq__(other)
253
254             2 из 20
255             def __len__(self):
256                 return len(self.bars)
257
258             2 из 20
259             def __iter__(self):
260                 return iter(self.bars)
261
262             2 из 20
263             def __getitem__(self, index):
264                 return self.bars[index]
265
266             2 из 20
267             def __repr__(self):
268                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
269
270             2 из 20
271             def __eq__(self, other):
272                 if not isinstance(other, FileStorage):
273                     return False
274                 if self.symbol != other.symbol:
275                     return False
276                 if self.timeframe != other.timeframe:
277                     return False
278                 if self.dt_from != other.dt_from:
279                     return False
280                 if self.dt_to != other.dt_to:
281                     return False
282                 return True
283
284             2 из 20
285             def __hash__(self):
286                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
287
288             2 из 20
289             def __ne__(self, other):
290                 return not self.__eq__(other)
291
292             2 из 20
293             def __len__(self):
294                 return len(self.bars)
295
296             2 из 20
297             def __iter__(self):
298                 return iter(self.bars)
299
300             2 из 20
301             def __getitem__(self, index):
302                 return self.bars[index]
303
304             2 из 20
305             def __repr__(self):
306                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
307
308             2 из 20
309             def __eq__(self, other):
310                 if not isinstance(other, FileStorage):
311                     return False
312                 if self.symbol != other.symbol:
313                     return False
314                 if self.timeframe != other.timeframe:
315                     return False
316                 if self.dt_from != other.dt_from:
317                     return False
318                 if self.dt_to != other.dt_to:
319                     return False
320                 return True
321
322             2 из 20
323             def __hash__(self):
324                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
325
326             2 из 20
327             def __ne__(self, other):
328                 return not self.__eq__(other)
329
330             2 из 20
331             def __len__(self):
332                 return len(self.bars)
333
334             2 из 20
335             def __iter__(self):
336                 return iter(self.bars)
337
338             2 из 20
339             def __getitem__(self, index):
340                 return self.bars[index]
341
342             2 из 20
343             def __repr__(self):
344                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
345
346             2 из 20
347             def __eq__(self, other):
348                 if not isinstance(other, FileStorage):
349                     return False
350                 if self.symbol != other.symbol:
351                     return False
352                 if self.timeframe != other.timeframe:
353                     return False
354                 if self.dt_from != other.dt_from:
355                     return False
356                 if self.dt_to != other.dt_to:
357                     return False
358                 return True
359
360             2 из 20
361             def __hash__(self):
362                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
363
364             2 из 20
365             def __ne__(self, other):
366                 return not self.__eq__(other)
367
368             2 из 20
369             def __len__(self):
370                 return len(self.bars)
371
372             2 из 20
373             def __iter__(self):
374                 return iter(self.bars)
375
376             2 из 20
377             def __getitem__(self, index):
378                 return self.bars[index]
379
380             2 из 20
381             def __repr__(self):
382                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
383
384             2 из 20
385             def __eq__(self, other):
386                 if not isinstance(other, FileStorage):
387                     return False
388                 if self.symbol != other.symbol:
389                     return False
390                 if self.timeframe != other.timeframe:
391                     return False
392                 if self.dt_from != other.dt_from:
393                     return False
394                 if self.dt_to != other.dt_to:
395                     return False
396                 return True
397
398             2 из 20
399             def __hash__(self):
400                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
401
402             2 из 20
403             def __ne__(self, other):
404                 return not self.__eq__(other)
405
406             2 из 20
407             def __len__(self):
408                 return len(self.bars)
409
410             2 из 20
411             def __iter__(self):
412                 return iter(self.bars)
413
414             2 из 20
415             def __getitem__(self, index):
416                 return self.bars[index]
417
418             2 из 20
419             def __repr__(self):
420                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
421
422             2 из 20
423             def __eq__(self, other):
424                 if not isinstance(other, FileStorage):
425                     return False
426                 if self.symbol != other.symbol:
427                     return False
428                 if self.timeframe != other.timeframe:
429                     return False
430                 if self.dt_from != other.dt_from:
431                     return False
432                 if self.dt_to != other.dt_to:
433                     return False
434                 return True
435
436             2 из 20
437             def __hash__(self):
438                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
439
440             2 из 20
441             def __ne__(self, other):
442                 return not self.__eq__(other)
443
444             2 из 20
445             def __len__(self):
446                 return len(self.bars)
447
448             2 из 20
449             def __iter__(self):
450                 return iter(self.bars)
451
452             2 из 20
453             def __getitem__(self, index):
454                 return self.bars[index]
455
456             2 из 20
457             def __repr__(self):
458                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
459
460             2 из 20
461             def __eq__(self, other):
462                 if not isinstance(other, FileStorage):
463                     return False
464                 if self.symbol != other.symbol:
465                     return False
466                 if self.timeframe != other.timeframe:
467                     return False
468                 if self.dt_from != other.dt_from:
469                     return False
470                 if self.dt_to != other.dt_to:
471                     return False
472                 return True
473
474             2 из 20
475             def __hash__(self):
476                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
477
478             2 из 20
479             def __ne__(self, other):
480                 return not self.__eq__(other)
481
482             2 из 20
483             def __len__(self):
484                 return len(self.bars)
485
486             2 из 20
487             def __iter__(self):
488                 return iter(self.bars)
489
490             2 из 20
491             def __getitem__(self, index):
492                 return self.bars[index]
493
494             2 из 20
495             def __repr__(self):
496                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
497
498             2 из 20
499             def __eq__(self, other):
500                 if not isinstance(other, FileStorage):
501                     return False
502                 if self.symbol != other.symbol:
503                     return False
504                 if self.timeframe != other.timeframe:
505                     return False
506                 if self.dt_from != other.dt_from:
507                     return False
508                 if self.dt_to != other.dt_to:
509                     return False
510                 return True
511
512             2 из 20
513             def __hash__(self):
514                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
515
516             2 из 20
517             def __ne__(self, other):
518                 return not self.__eq__(other)
519
520             2 из 20
521             def __len__(self):
522                 return len(self.bars)
523
524             2 из 20
525             def __iter__(self):
526                 return iter(self.bars)
527
528             2 из 20
529             def __getitem__(self, index):
530                 return self.bars[index]
531
532             2 из 20
533             def __repr__(self):
534                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
535
536             2 из 20
537             def __eq__(self, other):
538                 if not isinstance(other, FileStorage):
539                     return False
540                 if self.symbol != other.symbol:
541                     return False
542                 if self.timeframe != other.timeframe:
543                     return False
544                 if self.dt_from != other.dt_from:
545                     return False
546                 if self.dt_to != other.dt_to:
547                     return False
548                 return True
549
550             2 из 20
551             def __hash__(self):
552                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
553
554             2 из 20
555             def __ne__(self, other):
556                 return not self.__eq__(other)
557
558             2 из 20
559             def __len__(self):
560                 return len(self.bars)
561
562             2 из 20
563             def __iter__(self):
564                 return iter(self.bars)
565
566             2 из 20
567             def __getitem__(self, index):
568                 return self.bars[index]
569
570             2 из 20
571             def __repr__(self):
572                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
573
574             2 из 20
575             def __eq__(self, other):
576                 if not isinstance(other, FileStorage):
577                     return False
578                 if self.symbol != other.symbol:
579                     return False
580                 if self.timeframe != other.timeframe:
581                     return False
582                 if self.dt_from != other.dt_from:
583                     return False
584                 if self.dt_to != other.dt_to:
585                     return False
586                 return True
587
588             2 из 20
589             def __hash__(self):
590                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
591
592             2 из 20
593             def __ne__(self, other):
594                 return not self.__eq__(other)
595
596             2 из 20
597             def __len__(self):
598                 return len(self.bars)
599
600             2 из 20
601             def __iter__(self):
602                 return iter(self.bars)
603
604             2 из 20
605             def __getitem__(self, index):
606                 return self.bars[index]
607
608             2 из 20
609             def __repr__(self):
610                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
611
612             2 из 20
613             def __eq__(self, other):
614                 if not isinstance(other, FileStorage):
615                     return False
616                 if self.symbol != other.symbol:
617                     return False
618                 if self.timeframe != other.timeframe:
619                     return False
620                 if self.dt_from != other.dt_from:
621                     return False
622                 if self.dt_to != other.dt_to:
623                     return False
624                 return True
625
626             2 из 20
627             def __hash__(self):
628                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
629
630             2 из 20
631             def __ne__(self, other):
632                 return not self.__eq__(other)
633
634             2 из 20
635             def __len__(self):
636                 return len(self.bars)
637
638             2 из 20
639             def __iter__(self):
640                 return iter(self.bars)
641
642             2 из 20
643             def __getitem__(self, index):
644                 return self.bars[index]
645
646             2 из 20
647             def __repr__(self):
648                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
649
650             2 из 20
651             def __eq__(self, other):
652                 if not isinstance(other, FileStorage):
653                     return False
654                 if self.symbol != other.symbol:
655                     return False
656                 if self.timeframe != other.timeframe:
657                     return False
658                 if self.dt_from != other.dt_from:
659                     return False
660                 if self.dt_to != other.dt_to:
661                     return False
662                 return True
663
664             2 из 20
665             def __hash__(self):
666                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
667
668             2 из 20
669             def __ne__(self, other):
670                 return not self.__eq__(other)
671
672             2 из 20
673             def __len__(self):
674                 return len(self.bars)
675
676             2 из 20
677             def __iter__(self):
678                 return iter(self.bars)
679
680             2 из 20
681             def __getitem__(self, index):
682                 return self.bars[index]
683
684             2 из 20
685             def __repr__(self):
686                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
687
688             2 из 20
689             def __eq__(self, other):
690                 if not isinstance(other, FileStorage):
691                     return False
692                 if self.symbol != other.symbol:
693                     return False
694                 if self.timeframe != other.timeframe:
695                     return False
696                 if self.dt_from != other.dt_from:
697                     return False
698                 if self.dt_to != other.dt_to:
699                     return False
700                 return True
701
702             2 из 20
703             def __hash__(self):
704                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
705
706             2 из 20
707             def __ne__(self, other):
708                 return not self.__eq__(other)
709
710             2 из 20
711             def __len__(self):
712                 return len(self.bars)
713
714             2 из 20
715             def __iter__(self):
716                 return iter(self.bars)
717
718             2 из 20
719             def __getitem__(self, index):
720                 return self.bars[index]
721
722             2 из 20
723             def __repr__(self):
724                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
725
726             2 из 20
727             def __eq__(self, other):
728                 if not isinstance(other, FileStorage):
729                     return False
730                 if self.symbol != other.symbol:
731                     return False
732                 if self.timeframe != other.timeframe:
733                     return False
734                 if self.dt_from != other.dt_from:
735                     return False
736                 if self.dt_to != other.dt_to:
737                     return False
738                 return True
739
740             2 из 20
741             def __hash__(self):
742                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
743
744             2 из 20
745             def __ne__(self, other):
746                 return not self.__eq__(other)
747
748             2 из 20
749             def __len__(self):
750                 return len(self.bars)
751
752             2 из 20
753             def __iter__(self):
754                 return iter(self.bars)
755
756             2 из 20
757             def __getitem__(self, index):
758                 return self.bars[index]
759
760             2 из 20
761             def __repr__(self):
762                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
763
764             2 из 20
765             def __eq__(self, other):
766                 if not isinstance(other, FileStorage):
767                     return False
768                 if self.symbol != other.symbol:
769                     return False
770                 if self.timeframe != other.timeframe:
771                     return False
772                 if self.dt_from != other.dt_from:
773                     return False
774                 if self.dt_to != other.dt_to:
775                     return False
776                 return True
777
778             2 из 20
779             def __hash__(self):
780                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
781
782             2 из 20
783             def __ne__(self, other):
784                 return not self.__eq__(other)
785
786             2 из 20
787             def __len__(self):
788                 return len(self.bars)
789
790             2 из 20
791             def __iter__(self):
792                 return iter(self.bars)
793
794             2 из 20
795             def __getitem__(self, index):
796                 return self.bars[index]
797
798             2 из 20
799             def __repr__(self):
800                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
801
802             2 из 20
803             def __eq__(self, other):
804                 if not isinstance(other, FileStorage):
805                     return False
806                 if self.symbol != other.symbol:
807                     return False
808                 if self.timeframe != other.timeframe:
809                     return False
810                 if self.dt_from != other.dt_from:
811                     return False
812                 if self.dt_to != other.dt_to:
813                     return False
814                 return True
815
816             2 из 20
817             def __hash__(self):
818                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
819
820             2 из 20
821             def __ne__(self, other):
822                 return not self.__eq__(other)
823
824             2 из 20
825             def __len__(self):
826                 return len(self.bars)
827
828             2 из 20
829             def __iter__(self):
830                 return iter(self.bars)
831
832             2 из 20
833             def __getitem__(self, index):
834                 return self.bars[index]
835
836             2 из 20
837             def __repr__(self):
838                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
839
840             2 из 20
841             def __eq__(self, other):
842                 if not isinstance(other, FileStorage):
843                     return False
844                 if self.symbol != other.symbol:
845                     return False
846                 if self.timeframe != other.timeframe:
847                     return False
848                 if self.dt_from != other.dt_from:
849                     return False
850                 if self.dt_to != other.dt_to:
851                     return False
852                 return True
853
854             2 из 20
855             def __hash__(self):
856                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
857
858             2 из 20
859             def __ne__(self, other):
860                 return not self.__eq__(other)
861
862             2 из 20
863             def __len__(self):
864                 return len(self.bars)
865
866             2 из 20
867             def __iter__(self):
868                 return iter(self.bars)
869
870             2 из 20
871             def __getitem__(self, index):
872                 return self.bars[index]
873
874             2 из 20
875             def __repr__(self):
876                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
877
878             2 из 20
879             def __eq__(self, other):
880                 if not isinstance(other, FileStorage):
881                     return False
882                 if self.symbol != other.symbol:
883                     return False
884                 if self.timeframe != other.timeframe:
885                     return False
886                 if self.dt_from != other.dt_from:
887                     return False
888                 if self.dt_to != other.dt_to:
889                     return False
890                 return True
891
892             2 из 20
893             def __hash__(self):
894                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
895
896             2 из 20
897             def __ne__(self, other):
898                 return not self.__eq__(other)
899
900             2 из 20
901             def __len__(self):
902                 return len(self.bars)
903
904             2 из 20
905             def __iter__(self):
906                 return iter(self.bars)
907
908             2 из 20
909             def __getitem__(self, index):
910                 return self.bars[index]
911
912             2 из 20
913             def __repr__(self):
914                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
915
916             2 из 20
917             def __eq__(self, other):
918                 if not isinstance(other, FileStorage):
919                     return False
920                 if self.symbol != other.symbol:
921                     return False
922                 if self.timeframe != other.timeframe:
923                     return False
924                 if self.dt_from != other.dt_from:
925                     return False
926                 if self.dt_to != other.dt_to:
927                     return False
928                 return True
929
930             2 из 20
931             def __hash__(self):
932                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
933
934             2 из 20
935             def __ne__(self, other):
936                 return not self.__eq__(other)
937
938             2 из 20
939             def __len__(self):
940                 return len(self.bars)
941
942             2 из 20
943             def __iter__(self):
944                 return iter(self.bars)
945
946             2 из 20
947             def __getitem__(self, index):
948                 return self.bars[index]
949
950             2 из 20
951             def __repr__(self):
952                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
953
954             2 из 20
955             def __eq__(self, other):
956                 if not isinstance(other, FileStorage):
957                     return False
958                 if self.symbol != other.symbol:
959                     return False
960                 if self.timeframe != other.timeframe:
961                     return False
962                 if self.dt_from != other.dt_from:
963                     return False
964                 if self.dt_to != other.dt_to:
965                     return False
966                 return True
967
968             2 из 20
969             def __hash__(self):
970                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
971
972             2 из 20
973             def __ne__(self, other):
974                 return not self.__eq__(other)
975
976             2 из 20
977             def __len__(self):
978                 return len(self.bars)
979
980             2 из 20
981             def __iter__(self):
982                 return iter(self.bars)
983
984             2 из 20
985             def __getitem__(self, index):
986                 return self.bars[index]
987
988             2 из 20
989             def __repr__(self):
990                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
991
992             2 из 20
993             def __eq__(self, other):
994                 if not isinstance(other, FileStorage):
995                     return False
996                 if self.symbol != other.symbol:
997                     return False
998                 if self.timeframe != other.timeframe:
999                     return False
1000                if self.dt_from != other.dt_from:
1001                    return False
1002                if self.dt_to != other.dt_to:
1003                    return False
1004                return True
1005
1006             2 из 20
1007             def __hash__(self):
1008                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
1009
1010             2 из 20
1011             def __ne__(self, other):
1012                 return not self.__eq__(other)
1013
1014             2 из 20
1015             def __len__(self):
1016                 return len(self.bars)
1017
1018             2 из 20
1019             def __iter__(self):
1020                 return iter(self.bars)
1021
1022             2 из 20
1023             def __getitem__(self, index):
1024                 return self.bars[index]
1025
1026             2 из 20
1027             def __repr__(self):
1028                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
1029
1030             2 из 20
1031             def __eq__(self, other):
1032                 if not isinstance(other, FileStorage):
1033                     return False
1034                 if self.symbol != other.symbol:
1035                     return False
1036                 if self.timeframe != other.timeframe:
1037                     return False
1038                 if self.dt_from != other.dt_from:
1039                     return False
1040                 if self.dt_to != other.dt_to:
1041                     return False
1042                 return True
1043
1044             2 из 20
1045             def __hash__(self):
1046                 return hash((self.symbol, self.timeframe, self.dt_from, self.dt_to))
1047
1048             2 из 20
1049             def __ne__(self, other):
1050                 return not self.__eq__(other)
1051
1052             2 из 20
1053             def __len__(self):
1054                 return len(self.bars)
1055
1056             2 из 20
1057             def __iter__(self):
1058                 return iter(self.bars)
1059
1060             2 из 20
1061             def __getitem__(self, index):
1062                 return self.bars[index]
1063
1064             2 из 20
1065             def __repr__(self):
1066                 return f'FileStorage({self.symbol}, {self.timeframe}, {self.dt_from}, {self.dt_to})'
1067
1068             2 из 20
1069             def __eq__(self, other):
1070                 if not isinstance(other, FileStorage):
1071                     return False
1072                 if self.symbol != other.symbol:
1073                     return False
1074                 if self.timeframe != other.timeframe:
1075                     return False
1076                 if self.dt
```

- запуск одной стратегии на нескольких брокерах;
- диверсификация инфраструктурных рисков;
- удобство тестирования;
- возможность легко «мигрировать» между брокерами.

Переключение брокера = изменение параметра в конфиге.

Backtrader: стандарт де-факто, но с оговорками

Автор прямо говорит, что Backtrader – мощный инструмент, но он заброшен. В планах:

- либо создать собственный форк и привести архитектуру в порядок,
- либо полностью переписать движок,
- но сохранить совместимость со всеми существующими стратегиями.

До создания шаблона я тоже [пытался разработать собственный GUI для Backtrader](#) – простой интерфейс. Но проект особо не взлетел, хотя это и была попытка создать удобный шаблон, который каждый сможет расширять под себя.

Веб-интерфейс, gRPC и клиент-серверная архитектура

Логическим продолжением разговора стала архитектура будущей версии. Автор явно движется к полноценной платформе. Планы развития со слов Игоря выглядят так:

Веб-интерфейс. Для визуализации, анализа и, возможно, полноценной работы с роботами. Примерно как «домашний терминал».

gRPC. FinLabPy + роботы работают на VPS (сервер), а аналитика и управление – с локального ПК через gRPC-клиент. Это даст:

- безопасность,
- скорость,
- возможность разнести вычисления и интерфейс.

Из трёх часов стрима я отметил для себя несколько рекомендаций.

Практические советы разработчикам

2. Запускайте роботов только на VPS под Linux.

3. Используйте подробное логирование (DEBUG). Особенно в период отладки подписок и торговых операций.

4. Помните: унификация ограничена возможностями «самого слабого» брокера.

Если нужны уникальные функции – используйте методы конкретной обёртки напрямую.

Немного юридических моментов

Не обошлось и без юридических деталей – неожиданный, но важный момент, который автор пояснил. Автор планирует сертифицировать FinLabPy в реестре российского ПО Минцифры.

Это нужно:

1. для защиты авторских прав;
2. для потенциальной интеграции библиотеки брокерами или биржей.

Но для обычных трейдеров это вроде как никак не меняет ситуацию: проект планируется open-source.

Что пока остаётся «за кадром»

Как человек, который интересуется не только инфраструктурой, но и моделями, я отметил список тем, которые автор пока не стал раскрывать:

- продвинутые методы бэктестинга (walk-forward, Монте-Карло),
- управление рисками и портфельные модели,
- оптимизацию производительности,
- структуру конфигов,
- интеграцию LLM / AI в торговые системы.

Фокус шёл именно на инфраструктуру и унификацию.

Итог

Лично я воспринимаю [cia76/FinLabPy](#) как один из самых многообещающих open-source

проектов под российскую алготорговлю. Это попытка создать единый стандарт, которого

Автор: михаил шардин Моя онлайн-визитка Telegram «Умный Дом Инвестора»

18 ноября 2025

Игорь Чечет

торговые роботы

5.1K



15

46

29

**Михаил Шардин** Пермь 398

4 228

 с 23 января 2019 + Подписаться

46 КОММЕНТАРИЕВ

Сначала старые ▼



Beach Bunny

18 ноября 2025, 06:36

⋮

Ну судя по продаже им курсов за немаленькие деньги, информации в которых за такие деньги реально МАЛО и в желании продавать железку Raspberry Pi со своей системой попахивает инфоцыганством или желанием срубить бабла на лопатах.

„QUIK: использует свою экзотическую систему тикеров вида TQBR.SBER. “

Это формат Мосбиржи



+7



— Показать 4 ответа



Михаил

18 ноября 2025, 07:42

⋮

Когда вы реально напишите рабочий код на этой библиотеке, тогда и поговорим. На мой взгляд все такие библиотеки достаточно бессмысленны. Большинству не по силам разобраться в



Ilya Kosarev (kimkarus)

18 ноября 2025, 07:43

⋮

А как же osEngine на C# с открытым кодом?

— Показать 4 ответа

+1



T-800

18 ноября 2025, 08:46

⋮

Иногда проще написать свое, чем разбираться в чужих произведениях.

Просто в определенный момент времени можешь упереться в «не верное» решение в чужом произведении и будешь все время обставлять его костылями, в то время как в своем решении можно реализовать как считаешь правильным.

+10

Ещё 7 комментариев

Напишите комментарий...



ОТПРАВИТЬ

Читайте на SMART-LAB:



Формат QSH в OsEngine: поддержка и особенности работы

Недавно OsEngine начал поддержку бинарного формата хранения и трансляции данных по стаканам. Это было нужно, чтобы:...

OsEngine

18:01

Сегодня промышленность все чаще смотрит на ИТ как на инструмент для наращивания мощностей. Для российской...



Softline

17:16

Ресейл и поколение Z:
почему молодёжь выбирает
разумное потребление



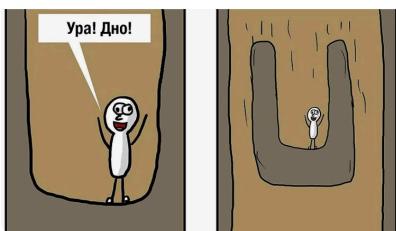
🧠 Ресейл и поколение Z: почему молодёжь выбирает разумное потребление

💻 Поколение Z относится к потреблению pragmatичнее, чем остальные. Для них важны не громкие слова и статус, а понятна...

Ⓜ️ МГКЛ

10:00

кабре идет ко дну - хуже не было никогда



Хэдхантер. Ситуация на рынке труда в декабре идет ко дну - хуже не было никогда

Вышла статистика рынка труда за декабрь 2025 года, которую Хэдхантер публикует ежемесячно, что же там интересного:...

Ⓜ️ Mozgovik

13.01.2026

Установите приложение Смартлаба:

RuStore

AppGallery

App Store



О смартлабе

Реклама

Полная версия



Московская Биржа является спонсором ресурса smart-lab.ru
Источник: ПАО Московская Биржа