УДК 004.42 Обзорная статья

https://doi.org/10.32603/2071-8985-2023-16-2-44-53

Аналитический терминал для тестирования торговых стратегий

А. В. Черевко, С. В. Букунов✉

Санкт-Петербургский государственный архитектурно-строительный университет, Санкт-Петербург, Россия

✉ sergeybukunov@yandex.ru

Аннотация. Описывается разработанное на кафедре информационных технологий Санкт-Петербургского архитектурно-строительного университета веб-приложение, предназначенное для тестирования торговых стратегий, используемых при проведении биржевых операций. Приложение может быть отнесено к классу программного обеспечения для бизнеса. Оно позволяет существенно повысить эффективность принятия инвестиционных решений как начинающими, так и профессиональными инвесторами. В приложении реализованы новый подход к написанию торговых стратегий с помощью шаблонов и возможность тестирования произвольного количества стратегий на инвестиционном портфеле, состоящем из произвольного количества различных финансовых активов. Приложение написано на языке Python с использованием библиотек Django и Django REST Framework. Для проведения тестирования используются данные по котировкам финансовых активов из открытых источников. Для хранения необходимой информации была спроектирована и реализована реляционная база данных (БД). Работа с базой данных осуществляется с помощью системы управления базами данных (СУБД) MySQL. Данные обрабатывались с помощью Python-библиотек numpy и pandas. Клиентская часть приложения реализована средствами языка TypeScript и его фреймворков React, React Router. Для работы с приложением не требуется установка дополнительного программного обеспечения, достаточно наличия доступа к сети Интернет.

Ключевые слова: торговые стратегии, инвестиционный портфель, веб-приложение, база данных, объектно-ориентированное программирование

Для цитирования: Черевко А. В., Букунов С. В. Аналитический терминал для тестирования торговых стратегий // Изв. СПбГЭТУ «ЛЭТИ». 2023. Т. 16, № 2. С. 44–53. doi: 10.32603/2071-8985-2023-16-2-44-53.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Original article

Analytical Terminal for Testing Trading Strategies

A. V. Cherevko, S. V. Bukunov✉

Saint Petersburg State University of Architecture and Civil Engineering, Saint Petersburg, Russia

✉ sergeybukunov@yandex.ru

Abstract. A web-application developed at the Department of Information Technologies of the Saint Petersburg State University of Architecture and Civil Engineering, which makes it possible to testing trading strategies used for stock exchange transactions, is described. The application can be categorized as business software. It allows you to significantly increase the efficiency of investment decision-making by both novice and professional investors. A new approach to writing trading strategies using templates is proposed. The ability to test an arbitrary number of strategies for an arbitrary structure investment portfolio is implemented. The application is developed by Python programming language and both Django and Django REST frameworks. Quotations data used for testing are taking from open sources. Relational data base was designed and implemented for data storage. MySQL data base management system (DBMS) is used to organize interaction with the database. The Python libraries numpy and pandas are used for data processing. The client part of the application is implemented using the TypeScript language and both React and React Router frameworks. The system does not re-

quire the installation of additional software. Access to the Internet need to use the system only. The approaches and technological solutions implemented in the application can be successfully used to create similar systems for other organizations and companies involved in similar activities.

Keywords: trading strategies, investment portfolio, web-application, data base, object-oriented programming

For citation: Cherevko A. V., Bukunov S. V. Analytical Terminal for Testing Trading Strategies // LETI Transactions on Electrical Engineering & Computer Science. 2023. Vol. 16, no. 2. P. 44–53. doi: 10.32603/2071-8985-2023-16-2-44-53.

Conflict of interest. The authors declare no conflicts of interest.

Введение. В настоящее время происходит активный рост числа людей, интересующихся сферой инвестиций. Так, по информации Московской биржи число физических лиц, имеющих на ней брокерские счета, в июне 2022 г. превысило 20 млн чел. [1]. Многие из них рассматривают эту сферу деятельности в качестве источника дополнительного дохода наряду с банковскими депозитами и вложениями в недвижимость. Однако сфера инвестиций традиционно относится к наиболее сложным видам деятельности [2] и требует не только определенного уровня экономических знаний, но и достаточно хорошей подготовки в области современных информационных технологий. Это связано с высоким уровнем компьютеризации и цифровизации всей финансовой сферы. В частности, вся биржевая торговля в настоящее время осуществляется через Интернет. Для подключения к торгам помимо Интернета необходимо соответствующее программное обеспечение, предоставляемое финансовой компанией, в которой частный инвестор открывает брокерский счет для проведения биржевых операций. В качестве таких компаний могут выступать банки, брокерские фирмы и другие организации, обладающие необходимыми лицензиями. Среди наиболее известных программных продуктов, предназначенных для работы на бирже, можно выделить следующие:

- MetaTrader, «MetaQuotes Software Corp» [3];
- Quik, «ARQA Technologies» [4];
- Transaq, «Скрин маркет системз» [5].

В последнее время для торговли на бирже широкое распространение получили так называемые торговые роботы [6], [7]. Торговый робот представляет собой компьютерную программу, которая совершает биржевые транзакции по определенному алгоритму. В основе любого алгоритма, в свою очередь, лежит определенная торговая стратегия (торговый метод) [8].

Необходимым этапом разработки торговой стратегии является ее тестирование на исторических данных по котировкам финансовых активов. Такое тестирование позволяет проверить работоспособность самой идеи, сформулированной в торговом методе, и тем самым уберечь инвестора от значительных убытков при использовании разработанной системы в реальных биржевых торгах. Поэтому многие (но не все) используемые в настоящее время торговые платформы содержат в себе опцию тестирования. Кроме того, такие возможности предоставляют и некоторые специализированные программные продукты, предназначенные для проведения аналитической обработки данных по финансовым рынкам. Среди программного обеспечения такого рода можно выделить, например, программные комплексы Metastock [9] и TSLab [10].

Среди недостатков перечисленных программных продуктов можно выделить следующие:

– для использования реальных торговых платформ необходимо открытие брокерского счета, а это не всегда необходимо на начальном этапе «вхождения» в инвестиционный процесс;

– для тестирования торговых стратегий во всех торговых платформах используются собственные встроенные языки программирования, освоение которых сопряжено с дополнительными трудозатратами;

– часть платформ, например TSLab [10], представляют собой визуальные конструкторы, позволяющие реализовывать только относительно несложные алгоритмы;

– некоторые платформы, например Metastock [9], относятся к разряду платного программного обеспечения.

Однако главный недостаток всех используемых платформ – это невозможность тестирования портфеля, состоящего из разных финансовых активов, по разным торговым стратегиям.

Постановка задачи. Цель настоящего исследования заключается в разработке веб-приложе-

ния, представляющего собой аналитический терминал, позволяющий с помощью сети Интернет получать информацию по котировкам финансовых активов, создавать собственные индикаторы для анализа финансовых активов, создавать и тестировать различные торговые стратегии на глубоко настраиваемых портфелях финансовых активов. Для реализации алгоритмов необходимо использовать популярный универсальный язык программирования Python. Приложение должно иметь единообразный, хорошо масштабируемый и производительный интерфейс, позволяющий не только эффективно добавлять новый функционал, но и модифицировать уже имеющийся.

Используемые технологии. Приложение разработано с помощью языка программирования Python [11] и его популярных фреймворков Django и Django REST Framework [12], [13] и реализует технологию «клиент–сервер».

Основная составляющая серверной части приложения – связка фреймворков Django и Django REST Framework. Использование архитектуры REST позволяет реализовать масштабируемость, прозрачность и единообразие интерфейса в сочетании с высокой производительностью, свойственным RESTful-приложениям.

Для связи между программой и сервером используется клиент-серверный протокол ASGI (Asynchronous Server Gateway Interface) на основе daphne – веб-сервера ASGI для Python [14].

Для взаимодействия с базой данных используется механизм объектно-ориентированного отображения (Object-Relation Mapper) [13]. Система ORM позволяет автоматизировать взаимодействие с базой данных и применяет объектно-ориентированный подход вместо SQL-инструкций. В качестве системы управления базой данных использовалась СУБД MySQL [15].

Для получения открытых рыночных данных по котировкам финансовых активов используется парсер на основе Python-библиотеки aiohttp.

В качестве инструментов для обработки данных используются Python-библиотеки numpy и pandas.

Клиентская часть приложения представляет собой пользовательский интерфейс, который формирует запросы к серверу и обрабатывает полученные от него ответы. При разработке этой части приложения использовался язык программирования TypeScript и его фреймворки React, React Router [16].

Архитектура приложения. Разработанное приложение строится по модульному принципу. Основные модули приложения:

• analysis – основной функциональный элемент, предназначенный для анализа созданных стратегий на созданных портфелях. Все результаты анализа сохраняются в виде записей (Log) для последующего рассмотрения;

• Quotes – основная база данных приложения. Содержит инструменты по мониторингу рынка ценных бумаг – получение списка ценных бумаг и их котировок, средства для расчета различных технических индикаторов и т. п.;

• Portfolio – модуль для создания и настройки различных портфелей. К настраиваемым параметрам относятся, например, баланс портфеля, состав портфеля, параметры для нерыночного закрытия позиций – так называемые стоп-приказы (sell stop/limit, buy stop/limit, stop loss, take profit), максимальное и минимальное количества одновременно открытых коротких или длинных позиций;

• Strategy – модуль для создания пользовательских стратегий;

• Log – модуль, отвечающий за хранение записей предыдущих анализов (Analysis). С его помощью можно просматривать и анализировать результаты работы различных стратегий.

UML-диаграмма компонентов приложения, отображающая его архитектуру, представлена на рис. 1.

Программная реализация приложения. Приложение реализовано на языке Python в объектно-ориентированном стиле. Каждый модуль приложения представляет собой совокупность некоторых сущностей, представляющих собой, как правило, Python-классы.

Модуль Quotes. Модуль состоит из двух основных сущностей: StockQuotes и StockInstance.

Класс StockQuotes содержит основные данные по рыночным инструментам:

– тикер (symbol) – краткое название котируемого инструмента, однозначно идентифицирующее его в системе;

– имя (name) – подробное имя рассматриваемого инструмента;

– котировки (quotes) – актуальные и исторические ценовые данные инструмента;

– тип (type) – тип финансового инструмента. На данный момент поддерживаются следующие типы рыночных инструментов: акция (stock), биржевой индексный фонд (ETF), облигация

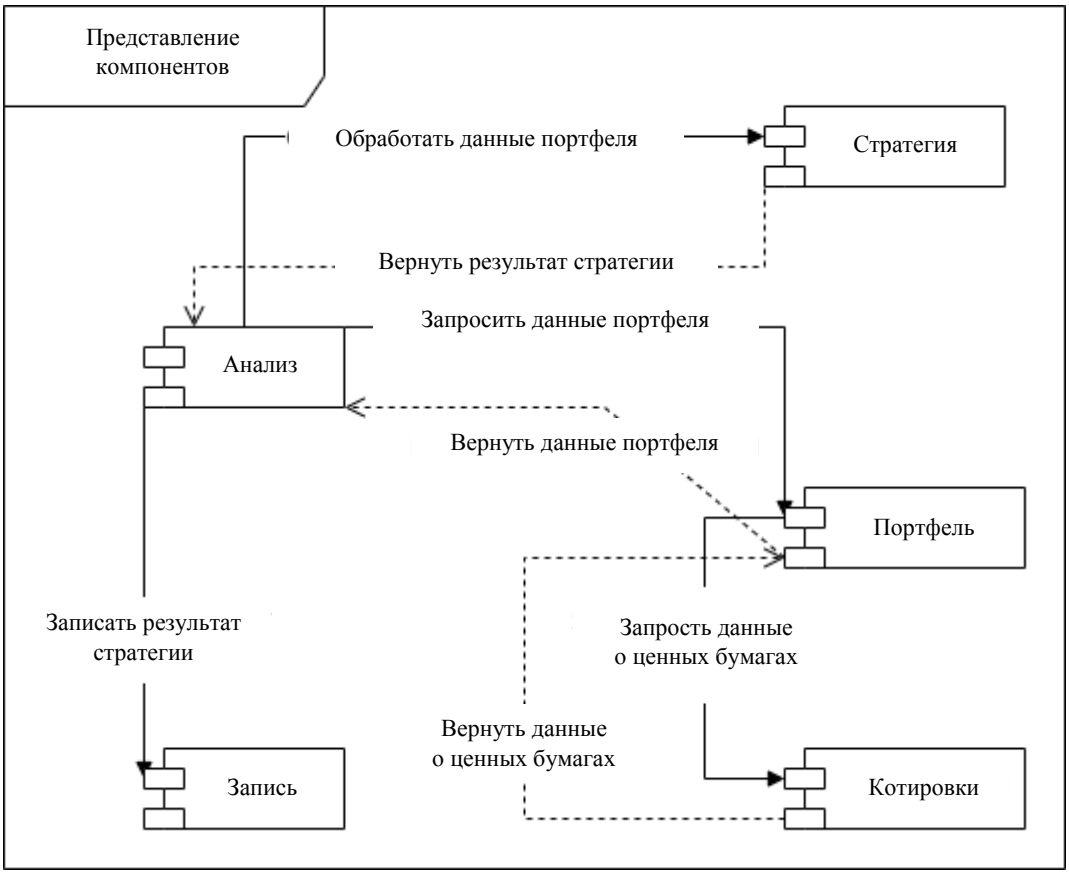

Рис.1. Диаграмма компонентов приложения
Fig. 1. Application Component Diagram

(bond), опцион (option), фьючерс (futures), валютная пара (FOREX);

– страна (country) – страна, к которой относится биржа, на которой торгуется данный инструмент;

– биржа (exchange) – название биржи, на которой торгуется данный инструмент.

Для работы с рассмотренными ранее данными в классе StockQuotes предназначены следующие методы:

• Обновить котировки (update_quotes) – совершает запрос к серверу с открытыми рыночными данными, для того чтобы актуализировать имеющиеся локальные данные.

• Получить тенденцию (get_tendency) – позволяет получить данные о последних изменениях цены актива, последние цены открытия, закрытия, максимальную и минимальную цены, а также объем торгов.

• Получить индикатор (get_indicator) – позволяет произвести расчет технического индикатора на основе имеющихся ценовых данных. В состав поддерживаемых на данный момент индикаторов входят наиболее популярные индикаторы, используемые при компьютерном анализе финансовых активов: любые виды скользящих средних (SMA, EMA, VWMA, а также любые WMA), индикатор схождения-расхождения скользящих средних (MACD), стохастик (Stochastic), балансовый объем (OBV) и полосы Боллинджера (Bollinger Bands).

Класс StockInstance необходим для хранения в портфеле различных ценных бумаг и содержит следующие атрибуты:

– котировки (quotes) – ссылка на соответствующий экземпляр сущности StockQuotes;

– количество (amount) – количество ценных бумаг соответствующего типа в портфеле;

– приоритет (priority) – текущий приоритет ценной бумаги в портфеле, определяющий порядок рассмотрения ее во всех списках и стратегиях.

Таким образом, компонент Quotes позволяет эффективно получать актуальные и исторические рыночные данные, а также рассчитывать на их основе различные технические индикаторы. В дополнение к этому Quotes позволяет привязывать к портфелям те или иные экземпляры ценных бумаг в виде их сущностей. При этом вся информация хранится в локальной SQL-базе данных, что открывает возможность для быстрого обращения к ней.

Модуль Portfolio. Модуль содержит единственную одноименную сущность – Portfolio, которая представляет собой модель инвестиционного портфеля, предоставляющую возможность для настройки и дальнейшего тестирования. Сущность содержит следующие атрибуты:

– имя (name) – символическое имя портфеля, отображаемое при его просмотре, а также служащее в качестве его вторичного идентификатора;

– баланс (balance) – текущий баланс брокерского счета, привязанного к портфелю;

– акции (stocks) – ценные бумаги в портфеле и их количество (StockInstance);

– Buy/Sell stop/limit – цена, при преодолении которой в соответствующем направлении осуществляется обязательная покупка или продажа достигшей ее ценной бумаги;

– Stop loss/Take profit – цена, при преодолении которой в соответствующем направлении осуществляется полное закрытие текущей позиции;

– Max/Min long/short (long/short_limit) – максимальное или минимальное количество одновременно открытых коротких или длинных позиций;

– создан/обновлен в последний раз (create/last_updated) – даты создания портфеля и последнего изменения его настроек.

Для обработки указанных данных класс Portfolio предоставляет следующие методы:

• Получить даты котировок (get_quotes_dates) – служит для получения временного промежутка, на котором котируются все хранящиеся на данный момент в портфеле ценные бумаги.

• Получить все котировки (get_all_quotes) – позволяет получить ценовые данные всех хранящихся на данный момент в портфеле ценных бумаг за указанный временной период. Имеется также возможность заполнения недостающих данных предшествующими им данными.

• Изменения цен ценных бумаг (stock_price_deltas) – необходим для получения величин изменения цен всех хранящихся на данный момент в портфеле ценных бумаг за указанный временной период.

• Рассчитать инвестиции (calculate_investment) – позволяет произвести расчет прибыли или убытков для данного портфеля за указанный период времени.

В результате модуль Portfolio позволяет создавать и гибко настраивать модели инвестиционных портфелей для расчета результатов инвестиций и дальнейшего тестирования с использованием различных стратегий.

Модуль Strategy. Модуль использует данные из модулей Portfolio и Quotes для расчетов результатов использования различных стратегий. При этом создание самой стратегии осуществляется в два этапа.

Этап 1. Все стратегии в данном модуле создаются в виде функций или функциональных объектов (классов с переопределенным методом __call__) в файле strategies.py, следующих определенному шаблону, что дает возможность очень удобно и просто добавлять новые стратегии к уже имеющимся.

Шаблон функции-стратегии на условном Python-подобном языке программирования выглядит следующим образом:

```
def <name>(
portfolio: Portfolio,
range_start: str | datetime.datetime | pandas.Timestamp | numpy.datetime64,
range_end: str | datetime.datetime | pandas.Timestamp | numpy.datetime64,
*args, **kwargs
) -> pandas.DataFrame:
logs = pandas.DataFrame(
    data={
    'balance': <balance*: float*>,
    'value': <value*: float*>,
    'stocks': <stocks: pandas.Series(
data=<amount*: int*>
index=<symbols*: str*>
)>
},
    index=<date_range: pandas.DateTimeIndex>
)
<calculations...>
return logs
```

Рассмотрим код представленного шаблона более подробно. Обязательными аргументами стратегии должны быть следующие сущности:

– portfolio – экземпляр модели портфеля, необходимый для получения всех его настроек, а также доступа к списку и котировкам ценных бумаг, хранящихся в нем на данный момент;

– range_start/range_end – строка в формате '<YYYY-MM-DD>' или объект одного из классов: datetime.datetime, pandas.Timestamp, numpy.datetime64. Представляет собой границы рассматриваемого временного промежутка.

Опционально можно добавлять и любые другие аргументы, которые в шаблоне указаны как *args, **kwargs. В качестве возвращаемого значения стратегия обязательно должна иметь объект класса pandas.DataFrame (т. е. таблицу) следующей структуры (по столбцам):

– ‘balance’ – данные о балансе портфеля (имеется в виду условный связанный с портфелем брокерский счет) в виде числа с плавающей точкой (float) на каждую отметку времени рассматриваемого промежутка;

– ‘value’ – данные о стоимости портфеля (баланс + цена закрытия всех имеющихся на данную временную отметку ценных бумаг) в виде числа с плавающей точкой (float) на каждую отметку времени рассматриваемого промежутка;

– ‘stocks’ – данные о количестве открытых позиций каждого вида (long/short) по каждому из типов имеющихся на данную временную отметку ценных бумаг в виде объекта класса pandas.Series, индексами которого выступают символы ценных бумаг, а данными – текущая позиция по ним.

В качестве индекса рассматриваемый фрейм данных должен иметь объект класса pandas.DateTimeIndex, представляющий собой набор всех временных отметок на рассматриваемом промежутке, причем частота отметок должна соответствовать рассматриваемому в стратегии таймфрейму. В случае необходимости валидации значений аргументов, передаваемых в стратегию, все проверки проводятся непосредственно внутри рассмотренной функции или функционального объекта.

Этап 2. После создания стратегии подобным образом (т. е. в виде шаблона) следует оформить ее клиентское представление. Для этого в файле strategies.py следует создать, если его не существует, список (list) с именем choices следующей структуры:

```
choices = [
    {
‘verbose_name’: <verbose_name: str>,
‘alias’: <alias: str>,
‘args’: {
‘arg_i’: <arg_i: Literal[‘int’, ‘float’, ‘str’, ‘list[int]’ , ‘list[float]’, ‘list[str]’], ...
}
}, ...
]
```

В качестве элементов списка choices выступают словари (dict) со следующим набором пар ключ–значение:

• ‘verbose_name’ – подробное имя стратегии в виде строки (str), которое будет отображаться при всех ее упоминаниях;

• ‘alias’ – псевдоним стратегии, ее уникальный идентификатор, обязательно совпадающий с именем соответствующей данной стратегии функции или функционального объекта в формате строки (str);

• ‘args’ – набор аргументов функции (функционального объекта), соответствующей(го) данной стратегии. Указываются лишь аргументы, не обязательные по умолчанию (все добавленные в процессе создания стратегии аргументы), т. е. все, кроме portfolio, range_start и range_end. Имя(ключ) и тип(значение) каждого аргумента обязательно должны соответствовать его имени и типу в аннотируемой стратегии. В качестве типа аргумента допускается использование только следующих строк (str):

– ‘int’ – целое число;

– ‘float’ – число с плавающей точкой;

– ‘str’ – строка;

– ‘list[int]’/‘list[float]’/’list[str]’ – список целых чисел/чисел с плавающей точкой/строк;

После выполнения двух рассмотренных этапов (создание функции-стратегии и ее клиентского представления по представленным шаблонам) стратегия готова к применению во всех остальных частях приложения.

Модуль Analysis. Модуль предназначен для тестирования имеющихся стратегий на созданных портфелях. Внутри данного модуля содержатся обработчики, с помощью которых данные из модулей Portfolio и Quotes передаются на обработку в модуль Strategy. В итоге создается запись, которая перенаправляется в модуль Log на сохранение. Возможность использования модуля Analysis появляется после создания хотя бы одного портфеля, удовлетворяющего базовым требованиям (ненулевой баланс, наличие хотя бы одной ценной бумаги), и хотя бы одной стратегии.

Модуль Log. Основу данного модуля составляет одноименная сущность, которая содержит в себе запись о результатах тестирования некоторого набора стратегий на каком-либо портфеле. В частности, сущность Log содержит следующие данные:

– range_start/range_end – границы временного промежутка тестирования;

– strategies – стратегии, принимавшие участие в тестировании;

– portfolio – инвестиционный портфель, принимавший участие в тестировании;

– logs – записи, возвращаемые стратегией и являющиеся непосредственно результатами тестирования.

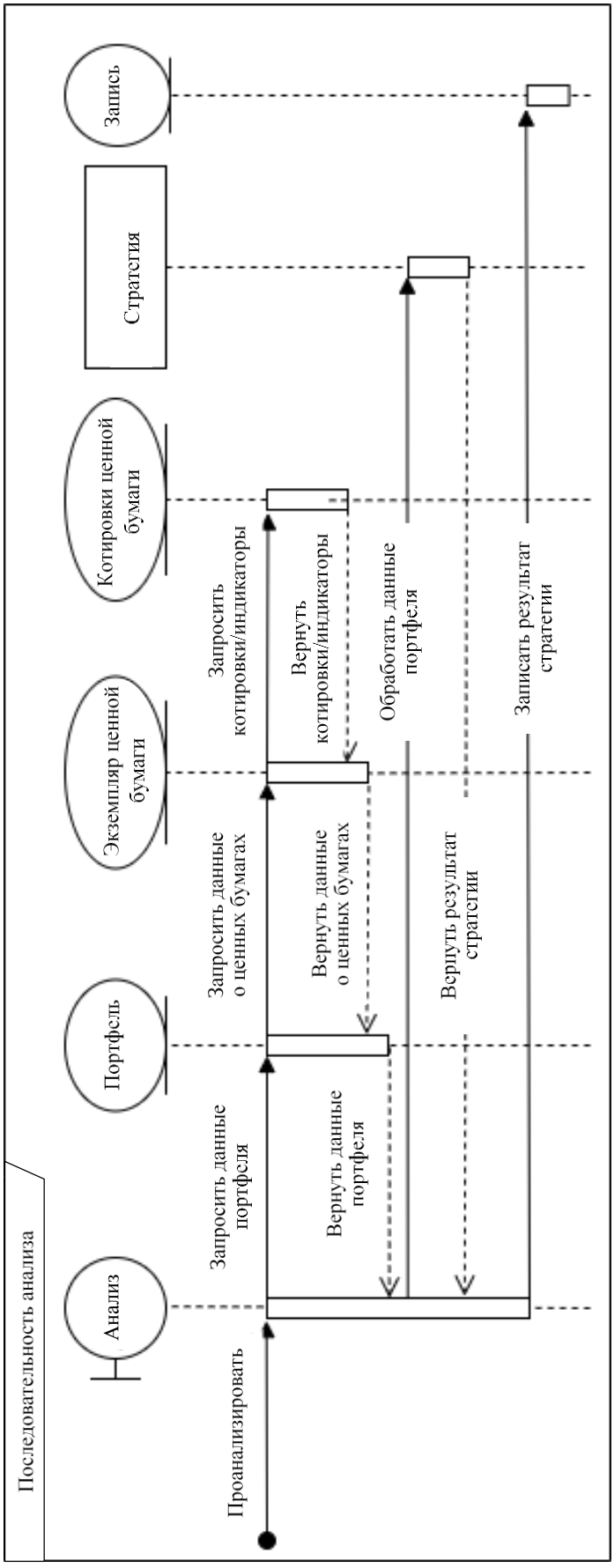

Рис.2. Диаграмма последовательности тестирования
Fig. 2. Testing Sequence Diagram

Рис. 3. Результаты тестирования
Fig. 3. Testing results

Результаты. Разработанное приложение позволяет проводить тестирование различных торговых стратегий на исторических данных по котировкам различных финансовых активов. На рис. 3 представлены результаты тестирования трех различных стратегий для акций компании Pulmonx

Corporation (тикер LUNG), торгуемых на бирже NASDAQ США.

На нижней диаграмме отображается динамика цен на акции компании (верхняя часть графика) и объемы совершенных сделок (нижняя часть графика) за период времени с 10.05.2021 по 04.02.2022. На верхней диаграмме отображается динамика балансовой стоимости портфеля для трех различных стратегий с названиями ma_levels (верхняя кривая), simple_periodic (средняя кривая) и mutual_ma_positions (нижняя кривая). Из представленных результатов, в частности, видно, что за заданный период времени цена на акции компании Pulmonx Corporation снизилась на 34.22 %. При этом использование стратегии simple_periodic позволило бы получить прибыль в размере 3.01 %, а использование стратегии ma_levels – прибыль в размере 4.53 %. Стратегия mutual_ma_positions в данном случае оказалась бы убыточной (убыток составил бы 6.14 %). Из полученных результатов можно сделать вывод, что для данного финансового актива предпочтительнее использовать стратегию ma_levels.

Заключение. С помощью библиотеки Django и языков программирования Python и TypeScript разработано веб-приложение относящееся к классу бизнес-приложений и предназначенное для создания и тестирования различных торговых стратегий перед их использованием в реальных биржевых операциях. При разработке стратегий можно использовать основные индикаторы компьютерного анализа, реализованные в приложении. Кроме того, приложение позволяет создавать свои собственные индикаторы. Отличительной особенностью приложения является реализация единого шаблона для описания различных стратегий. Новизна работы заключается еще и в том, что в отличие от всех существующих на сегодняшний день торговых платформ разработанное приложение позволяет проводить тестирование произвольного количества стратегий на инвестиционном портфеле, состоящем из произвольного количества финансовых инструментов, что может существенно повысить эффективность принятия инвестиционных решений как начинающими, так и профессиональными инвесторами.

Список литературы

1. Количество частных инвесторов на Московской бирже превысило 20 миллионов. URL: https://www.moex.com/n49811/?nt=106 (дата обращения 10.07.2022).

2. Sharpe W. F., Alexander G. J., Baily J. V. Investments. New Jersy: Prentice Hall, 1998. 990 p.

3. Торговая платформа Metatrader5. URL: https://metatrader5.com/ru/trading-platform (дата обращения 15.07.2022).

4. Программный комплекс QUIK. URL: https://arqatech.com/ru/products/quik (дата обращения 15.07.2022).

5. Система брокерского обслуживания Transaq. URL: https://transaq.ru (дата обращения 15.07.2022).

6. Чеботарев Ю. А. Торговые роботы на российском фондовом рынке. М.: СмартБук, 2011. 160 с.

7. Роботы помогли Мосбирже поставить рекорд в марте. URL: https://1prime.ru/Financial_market/20200417/831279655.html (дата обращения 18.07.2022).

8. Солабуто Н. В. Трейдинг: торговые системы и методы. СПб.: Питер, 2010. 336 с.

9. Metastock. URL: https://www.metastock.com/ (дата обращения 15.11.2022).

10. Торговая платформа TSLab. URL: https://doc.tslab.pro/tslab/ (дата обращения 25.11.2022).

11. Златопольский Д. Основы программирования на языке Python. М.: ДМК-Пресс, 2018. 396 с.

12. Дронов В. А. Django 2.1. Практика создания веб-сайтов на Python. СПб.: БХВ-Петербург, 2019. 672 с.

13. Меле А. Django 2 в примерах. М.: ДМК-Пресс, 2019. 408 с.

14. Daphne 0.8.1. An ASGI web server for Python. URL: https://pypi.org/project/daphne/0.8.1/ (дата обращения 05.09.2022).

15. Шварц Б., Ткаченко В., Зайцев П. MySQL по максимуму. Оптимизация, репликация, резервное копирование. СПб.: Питер, 2018. 336 с.

16. Черный Б. Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений. СПб.: Питер, 2020. 352 с.

Информация об авторах

Черевко Антон Валерьевич – студент Санкт-Петербургского государственного архитектурно-строительного университета, ул. 2-я Красноармейская, д. 4, Санкт-Петербург, 190005, Россия.
E-mail: antonycherevko@gmail.com

Букунов Сергей Витальевич – канд. техн. наук, доцент. Санкт-Петербургский государственный архитектурно-строительный университет, ул. 2-я Красноармейская, д. 4, Санкт-Петербург, 190005, Россия.
E-mail: sergeybukunov@yandex.ru
http://orcid.org/0000-0002-5983-0637

References

1. Kolichestvo chastny`x investorov na Moskovskoj birzhe prevy`silo 20 millionov. URL: https://www.moex.com/n49811/?nt=106 (data obrashheniya 10.07.2022). (In Russ.).

2. Sharpe W. F., Alexander G. J., Baily J. V. Investments. New Jersy: Prentice Hall, 1998. 990 p.

3. Torgovaya platforma Metatrader5. URL: https://metatrader5.com/ru/trading-platform (data obrashheniya 15.07.2022). (In Russ.).

4. Programmny`j kompleks QUIK. URL: https://arqatech.com/ru/products/quik (data obrashheniya 15.07.2022). (In Russ.).

5. Sistema brokerskogo obsluzhivaniya Transaq. URL: https://transaq.ru (data obrashheniya 15.07.2022). (In Russ.).

6. Chebotarev Yu. A. Torgovy`e roboty` na rossijskom fondovom ry`nke. M: SmartBuk, 2011. 160 s. (In Russ.).

7. Roboty` pomogli Mosbirzhe postavit` rekord v marte. URL: https://1prime.ru/Financial_market/20200417/831279655.html (data obrashheniya 18.04.2022). (In Russ.).

8. Solabuto N. V. Trejding: torgovy`e sistemy` i metody`. SPb.: Piter, 2010. 336sc. (In Russ.).

9. Metastock. URL: https://www.metastock.com/ (data obrashheniya 15.11.2022).

10. Torgovaya platforma TSLab. URL: https://doc.tslab.pro/tslab/ (data obrashheniya 25.11.2022).

11. Zlatopol`skij D. Osnovy` programmirovaniya na yazy`ke Python. M.: DMK-Press, 2018. 396 s. (In Russ.).

12. Dronov V. A. Django 2.1. Praktika sozdaniya vebsajtov na Python. SPb.: BXV-Peterburg, 2019. 672 s. (In Russ.).

13. Mele A. Django 2 v primerax. M.: DMK-Press, 2019. 408 s. (In Russ.).

14. Daphne 0.8.1. An ASGI web server for Python. URL: https://pypi.org/project/daphne/0.8.1/ (data obrashheniya 05.09.2022).

15. Shvarcz B., Tkachenko V., Zajcev P. MySQL po maksimumu. Optimizaciya, replikaciya, rezervnoe kopirovanie. SPb.: Piter, 2018. 336 s. (In Russ.).

16. Cherny`j B. Professional`ny`j TypeScript. Razrabotka masshtabiruemy`x JavScript-prilozhenij. SPb.: Piter, 2020. 352 s. (In Russ.).

Information about the authors

Anton V. Cherevko – student of Saint Petersburg State University of Architecture and Civil Engineering, 2nd Krasnoarmeiskaya str., 4, Saint Petersburg, 190005, Russia.
E-mail: antonycherevko@gmail.com

Sergey V. Bukunov – Cand. Sci. (Eng.), Assistant Professor of Saint-Petersburg State University of Architecture and Civil Engineering, 2nd Krasnoarmeiskaya str., 4, Saint Petersburg, 190005, Russia.
E-mail: sergeybukunov@yandex.ru
http://orcid.org/0000-0002-5983-0637

Статья поступила в редакцию 20.12.2022; принята к публикации после рецензирования 29.12.2022; опубликована онлайн 28.02.2023.

Submitted 20.12.2022; accepted 29.12.2022; published online 28.02.2023.