Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Системное программирование

Лавров Владимир Александрович

Проектирование и разработка платформы для портфельного менеджмента

Бакалаврская работа

Научный руководитель:
доц. кафедры СП, канд. физ.-мат. наук К. Ю. Романовский

Рецензент:
Менеджер проектов,
Филиал Частной акционерной компании с ограниченной ответственностью «ДиЭсЭкс ТЕХНОЛОДЖИЗ ЛИМИТЕД» ДиЭсЭкс Технолоджиз Раша Р. В. Белков

Санкт-Петербург
2020

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
System Programming

Vladimir Lavrov

Design and development of a portfolio management platform

Graduation Thesis

Scientific supervisor:
Associate professor, PhD (CSc.) Konstantin Romanovsky

Reviewer:
Project Manager, DSX Technologies Limited Roman Belkov

Saint-Petersburg
2020

Оглавление

Введение

В последнее время всё бóльшую популярность приобретают онлайн платформы для торговли и инвестиций. Всё больше людей учатся финансовой грамотности и начинают торговать ценными бумагами, потому что это быстро, удобно и не требует большого стартового капитала. Следовательно повышается спрос на различные приложения, позволяющие упростить процесс торговли. О разработке такого приложения и пойдет речь в этой работе.

Инвесторы оперируют понятием «портфель активов» или, по-другому, «инвестиционный портфель». Инвестиционный портфель — это совокупность активов (валют, ценных бумаг и так далее...), управляемая инвестором как единое целое. У одного инвестора может быть несколько портфелей, каждый портфель создается с какой-то целью. Например, у инвестора может быть так называемый «агрессивный» портфель, сформированный из активов с высокой доходностью и высоким уровнем риска, и «пассивный» портфель, ориентированный на стабильный доход на уровне рыночного индекса.

Состояние инвестиционного портфеля определяется набором сделок и транзакций, совершённых в рамках этого портфеля. Сделка — операция по обмену определенного количества одного актива на определенное количество другого актива. Например, один инвестор готов продать N акций по цене X в валюте C. Если находится другой инвестор, готовый купить N акций по цене X в валюте C, совершается сделка, и они обмениваются активами. Транзакция — операция по списанию или зачислению определенного количества одного актива в инвестиционный портфель. Например, когда портфель только создан, в нем нет ресурсов (активов) для совершения сделок, поэтому необходимо совершить транзакцию на ввод средств, на которые впоследствии будут приобретаться, например, акции.

Портфельный менеджмент - процесс управления инвестиционным портфелем. Этот процесс можно значительно облегчить, если использовать приложение, позволяющее отображать финансовые операции

пользователя в удобной и компактной форме, а также показывать в какой-либо форме успешность его торговли, например, строить графики, основываясь на имеющихся у пользователя активах и изменении их стоимости с течением времени, или автоматически вычислять различные метрики по портфелю, позволяющие принимать более взвешенные решения.

Рассмотрим конкретный пример, когда приложение позволяет упростить портфельный менеджмент. Одно из наиболее важных и популярных свойств портфеля — диверсификация. Диверсификация — это распределение активов в портфеле между различными объектами вложений с целью снижения риска возможных потерь капитала. Одно дело, когда пользователь определяет степень диверсификации портфеля, основываясь на некоторых характеристиках компаний, активы которых лежат в этом портфеле. Другое дело, когда программа, основываясь на исторических данных о котировках активов, лежащих в портфеле, выдает численную характеристику диверсификации портфеля. Второй способ предпочтительнее, так как он позволяет, например, сравнить диверсификацию двух разных портфелей. Задача приложения, создаваемого в рамках этой работы, — упростить анализ торговли пользователя за счёт предоставления ему различных графиков и метрик по портфелю.

1. Постановка задачи

Целью данной дипломной работы является проектирование и разработка серверной части продукта для портфельного менеджмента. Для достижения этой цели были сформулированы следующие задачи.

- Проанализировать существующие приложения для портфельного менеджмента
- Выявить требования к серверной части продукта
- Спроектировать серверную часть продукта
- Реализовать серверную часть продукта
- Развернуть и протестировать серверную часть продукта

2. Анализ существующих приложений

Для начала выделим критерии для сравнения рассматриваемых приложений. Сделаем упор на наличие разных типов метрик по портфелю и на удобство использования приложения.

2.1. Критерии для сравнения

1. Наличие базовых метрик по портфелю
   Список базовых метрик:

   - График стоимости конкретного актива в базовой валюте (основной валюте портфеля, выбранной пользователем)
   - График стоимости портфеля в базовой валюте
   - Круговая диаграмма распределения активов в портфеле

2. Наличие нетривиальных метрик по портфелю
   Список нетривиальных метрик:

   - График доходности конкретного актива
     Нетривиальность этой метрики заключается в том, что существует несколько алгоритмов её подсчёта, при этом ни один из них не является абсолютно правильным (подробности про способы подсчёта этой метрики можно посмотреть в блоге [3] от разработчиков сервиса для портфельного менеджмента Intelinvest). Предоставление пользователю возможности самому выбрать алгоритм подсчета может привести к проблеме перегрузки интерфейса, а выбор конкретного алгоритма — к некорректному подсчёту.
   - График доходности портфеля
     При подсчёте доходности портфеля возникают те же проблемы, что и при подсчёте доходности одного конкретного актива.

- График сравнения доходности портфеля с уровнем фондового индекса и уровнем инфляции
  Интересная для пользователя и тривиальная для подсчёта метрика, однако она требует подсчета доходности портфеля.
- График корреляции выбранных активов
  Эта метрика помогает определить степень диверсификации портфеля.

3. Поддержка API торговых площадок для автоматической загрузки финансовых операций пользователя

4. Кроссплатформенность
   В данном случае подразумевается наличие ios, android и web/desktop версий приложения.

5. Низкий порог вхождения
   Важно, чтобы приложение было интересно для новичков и любителей, которые не занимаются трейдингом постоянно. Подразумевается, что в приложении нет перегруженного интерфейса, заточенного под профессионалов.

Рассмотрим существующие на рынке аналоги нашего приложения. Будем рассматривать приложения, позволяющие хранить и анализировать информацию об инвестиционном портфеле.

2.2. Существующие приложения

- Blockfolio [5]
  Blockfolio позволяет получить информацию по криптовалютному портфелю и следить за его динамикой. Также Blockfolio позволяет быть в курсе всех важных событий — приложение агрегирует новости, связанные с криптовалютой, из различных источников. Приложение интегрируется с более чем 60 биржами, поддерживает механизм оповещений при достижении криптовалютой определенной цены на той или иной бирже.

- Personal Capital [10]
  Personal Capital позволяет видеть все счета пользователя (банковские счета, пенсионные фонды и инвестиции) в одном месте. После того как пользователь ввёл информацию о всех необходимых счетах, Personal Capital будет следить за изменениями статуса и сообщать о новых списаниях либо поступлениях. Personal Capital работает с большинством банков и кредитных агенств в США, а также со всеми основными инвестиционными брокерами.

- Investment Account Manager [9]
  Investment Account Manager позволяет получить подробную информацию по инвестиционному портфелю. Достоинство этого приложения — возможность генерировать более 20 разных типов отчетов по итогам торговли в рамках портфеля. Investment Account Manager предоставляет возможность подсчитать прогнозируемую прибыль по портфелю и предлагает более выгодную перебалансировку активов в портфеле.

- Intelinvest [7]
  Intelinvest — приложение для управления инвестиционным порт-

фелем, ориентированное на российский рынок. Поддерживает автоматический импорт отчетов от большинства популярных брокеров. Позволяет подсчитать прибыль и доходность по портфелю, график его стоимости, отобразить круговую диаграмму соотношения стоимости активов внутри портфеля и некоторые другие аналитические показатели.

2.3. Таблица сравнения

Таблица 1: Сравнение приложений по выделенным критериям

Приложение	Баз. метр.	Нетр. метр.	API	Кросспл.	Порог вх.
Blockfolio	+	−	+	+	+
Personal Capital	+	−	+	+	+
IAM	+	+−	−	−	−
Intelinvest	+	−+	+	+−	+

Стоит отметить, что, например, разработчики приложения Intelinvest стараются добавлять некоторые нетривиальные метрики. Они добавили график доходности портфеля, но получают много негативных комментариев по поводу некорректного подсчета доходности.

Это далеко не все существующие приложения, позволяющие анализировать инвестиционный портфель, но одни из наиболее популярных. В этой таблице хотелось бы отметить следующую тенденцию: все современные кроссплатформенные приложения с дружелюбным для пользователя интерфейсом умеют подсчитывать только базовый набор метрик по портфелю, в то время как приложение Investment Account Manager, позволяющее отобразить большое количество отчётов по итогам торговли, имеет только desktop-версию, которой недостаточно для большинства современных пользователей, которые привыкли работать с мобильными устройствами (согласно исследованиям компании Criteo, описанным в источниках [6] и [11]). К тому же она имеет перегруженный интерфейс, заточенный под профессионалов.

3. Требования к серверной части продукта

По результатам обсуждений наша команда пришла к выводу, что к серверной части продукта предъявляются следующие требования.

3.1. Функциональные требования

- Регистрация и авторизация пользователей
- Расчет и хранение данных о портфелях
  Поддержка следующих операций:
  - Создание портфеля
  - Получение данных о портфеле
  - Изменение данных о портфеле
  - Удаление портфеля
- Расчет и хранение данных о торговой деятельности
  Поддержка следующих операций:
  - Добавление сделок/транзакций в портфель
  - Получение данных о сделках/транзакциях в рамках портфеля
  - Изменение данных о сделках/транзакциях в рамках портфеля
  - Удаление сделок/транзакций из портфеля
- Импорт данных о торговой деятельности из сторонних источников
  Поддержка следующих способов автоматического добавления сделок/транзакций.
  - Импорт из файлов (формат csv)
  - Импорт с помощью программного интерфейса торговых площадок

- Хранение и загрузка из сторонних источников данных об истории котировок валют и других активов

3.2. Нефункциональные требования

- Масштабируемость
  В данном случае подразумевается горизонтальная масштабируемость.

- Хранение истории котировок по 1000 активов с интервалом в один день, предусмотреть возможность увеличения количества активов до 50000 и возможность хранения котировок с более низкой дискретностью
  Было принято решение хранить историю котировок активов внутри разрабатываемой системы по следующим причинам:

  1. Во внешних сервисах есть ограничения на количество запросов в секунду/минуту/день и так далее... Прослойка в виде отдельного сервиса между приложениями-клиентами и внешними сервисами, предоставляющими данные о котировках, позволяет достаточно просто учитывать ограничения на количество запросов.
  2. Для загрузки котировок активов необходимо подключаться к нескольким сервисам, так как не существует универсального сервиса, предоставляющего котировки по всем интересующим нас активам. Разные сервисы предоставляют разные интерфейсы, а прослойка в виде отдельного сервиса позволяет унифицировать интерфейс для приложений-клиентов. Такой подход позволяет написать код, отвечающий за преобразование данных их форматов сторонних источников в формат проекта, всего один раз, а не дублировать его во всех приложениях-клиентах.
  3. Внешние сервисы могут быть недоступны или могут внезапно менять API, что автоматически выведет из строя большую

часть функциональности приложения.

- Документация и примеры использования API, поддержание их актуальности
  Для того чтобы упростить интеграцию с приложениями-клиентами, выдвинуто требование о документации всех точек входа API. Подробная документация также позволит упростить разработку новых приложений-клиентов.

- Поддержка интерфейса двух торговых площадок
  В рамках этой работы выдвинуто требование о поддержке возможности импорта данных о торговой деятельности пользователя из двух торговых площадок: DSX Global и Тинькофф Инвестиции.

4. Проектирование серверной части продукта

Для разработки всего продукта, включая приложения-клиенты, была выбрана трехуровневая архитектура (Рис. 1), которая на данный момент является стандартом при разработке бизнес-приложений [12].

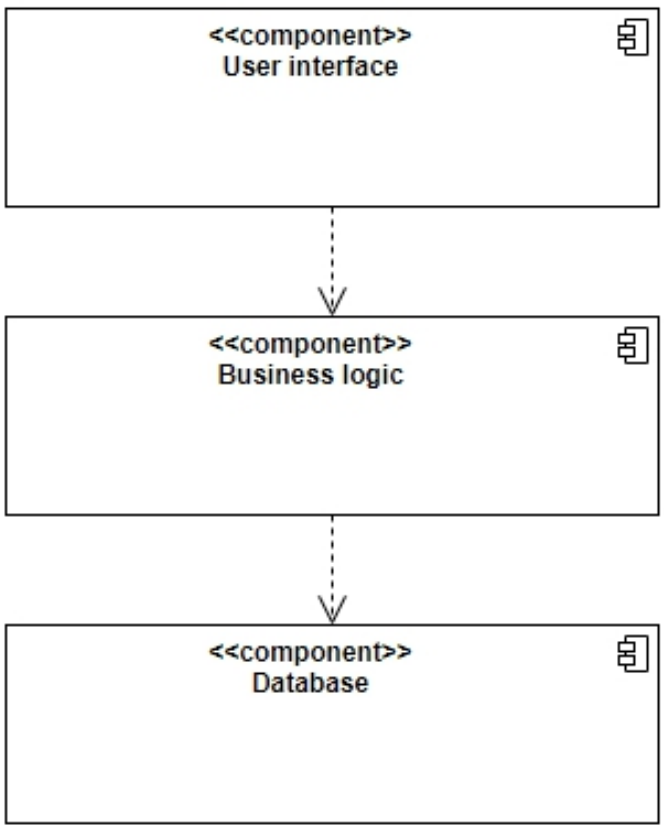

Рис. 1: Трехуровневая архитектура

По сравнению с двухуровневой моделью она обеспечивает:

- Бóльшую масштабируемость
  Масштабируемость обеспечивается за счет возможности одновременной работы нескольких экземпляров приложения, которое отвечает за слой бизнес-логики. Возможность одновременной работы нескольких экземпляров приложения, в свою очередь, обеспечивается за счет изоляции слоя бизнес-логики от слоя данных.

- Бóльшую конфигурируемость

  Конфигурируемость обеспечивается также за счет изоляции слоя бизнес-логики от слоя данных. Например, замена СУБД в таком случае требует намного меньше изменений в коде программы.

Итак, в этой работе рассматриваем слой бизнес-логики и слой данных в рамках трехуровневой модели разработки бизнес приложений. Приложение можно условно разделить на два модуля. Первый модуль, который будем называть portfolio service, отвечает за хранение и обработку данных о пользователях и их портфелях, второй, который будем называть marketdata provider, — за загрузку и хранение данных о котировках активов. Было принято решение создать отдельный restful сервис для каждого модуля (Рис. 2), так как модуль marketdata provider не зависит от модуля portfolio service и в дальнейшем может быть переиспользован в других проектах.

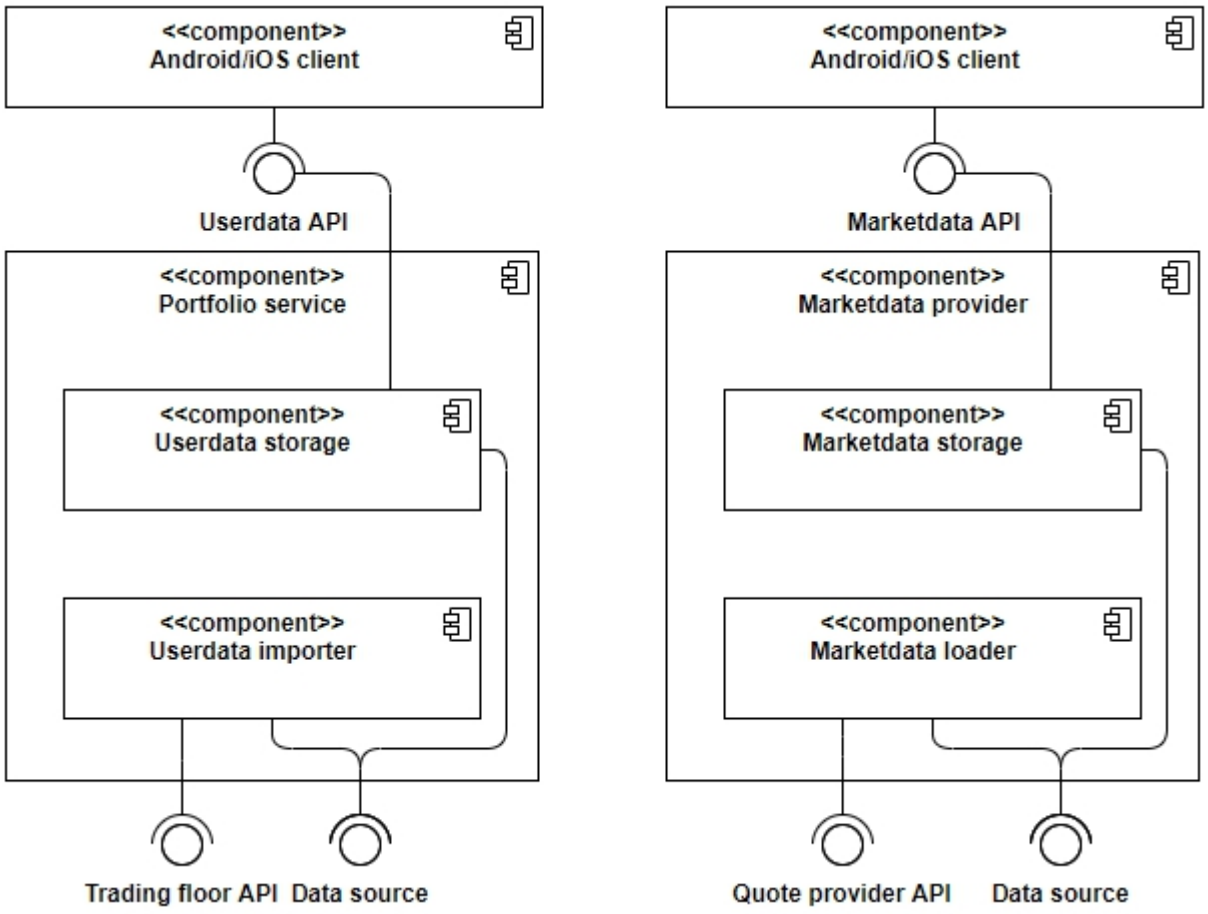

Рис. 2: Архитектура слоя бизнес-логики трехуровневой модели

Итак, рассмотрим подробнее архитектуру каждого сервиса. Portfolio service имеет слоистую архитектуру (Рис. 3), которая является стандартом в подобного рода приложениях. Он содержит следующие слои:

- Слой контроллеров
  Слой контроллеров предназначен для связи компонента пользовательского интерфейса и компонента бизнес-логики трехуровневой модели. В этом слое определяется API для работы с серверной частью приложения.

- Слой бизнес-логики
  Слой бизнес-логики предназначен для моделирования предметной области. В данном случае реализована анемичная модель предметной области, то есть все действия, ассоциированные с объектами предметной области, вынесены в отдельные сервисы. Это позволяет упростить код в случае, когда нет сложной логики взаимодействия объектов (в нашем примере как раз такой случай), а также упростить сериализацию объектов.

- Слой репозиториев
  Слой репозиториев предназначен для интеграции приложения с базой данных.

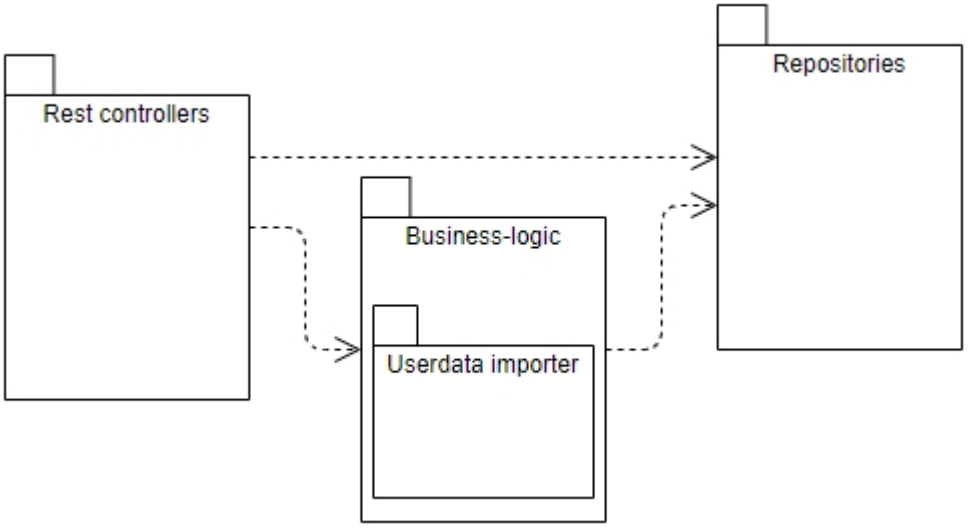

Рис. 3: Архитектура portfolio service

В слое бизнес-логики на рис. 3 изображён компонент Userdata importer, отвечающий за импорт данных о финансовой деятельности пользователей из сторонних торговых площадок. Рассмотрим его архитектуру. Согласно требованиям надо реализовать два способа импорта данных: из csv файла и с помощью API торговых площадок. На рис. 4 показана диаграмма классов, связанных с импортом данных из csv файла, а на рис. 5 — аналогичная ей диаграмма классов, связанных с импортом данных через API торговых площадок. Большинство торговых площадок позволяют экспортировать данные о сделках и транзакциях пользователей в csv файл, при этом у каждой площадки формат этого файла свой. Для каждой торговой площадки создаем свой класс, отвечающий за преобразование данных о сделках и транзакциях из формата данной торговой площадки в формат, принятый в проекте. Затем объединяем их общим интерфейсом, для того чтобы сложить в репозиторий. В репозитории объекты хранятся в словаре, где ключ — название торговой площадки. В дальнейшем планируется вынести этот компонент в отдельную библиотеку для переиспользования в других проектах.

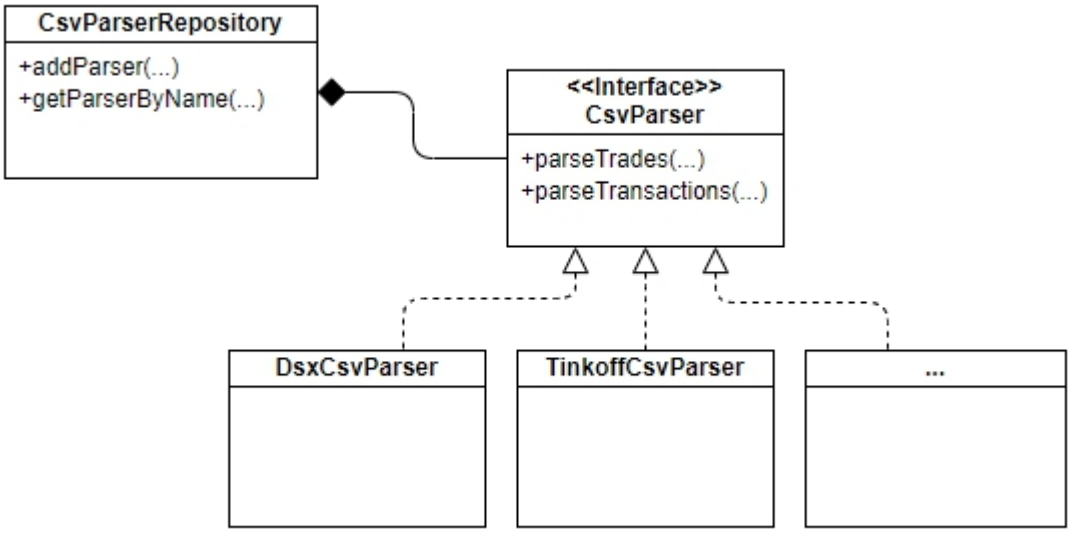

Рис. 4: Модуль, отвечающий за импорт данных из csv файла

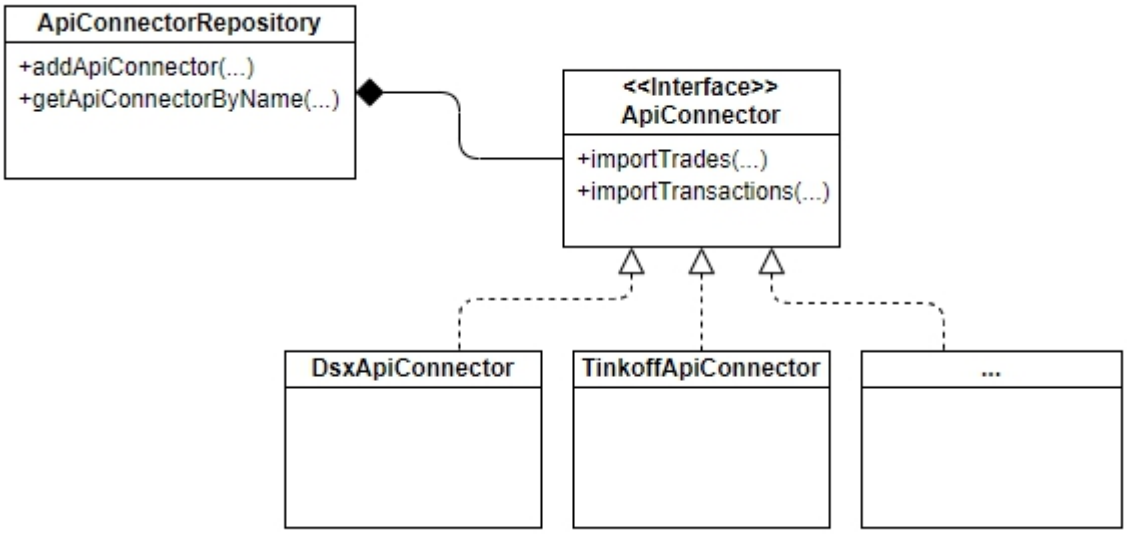

Рис. 5: Модуль, отвечающий за импорт данных с помощью API торговых площадок

Marketdata provider имеет аналогичную portfolio service слоистую архитектуру (Рис. 6).

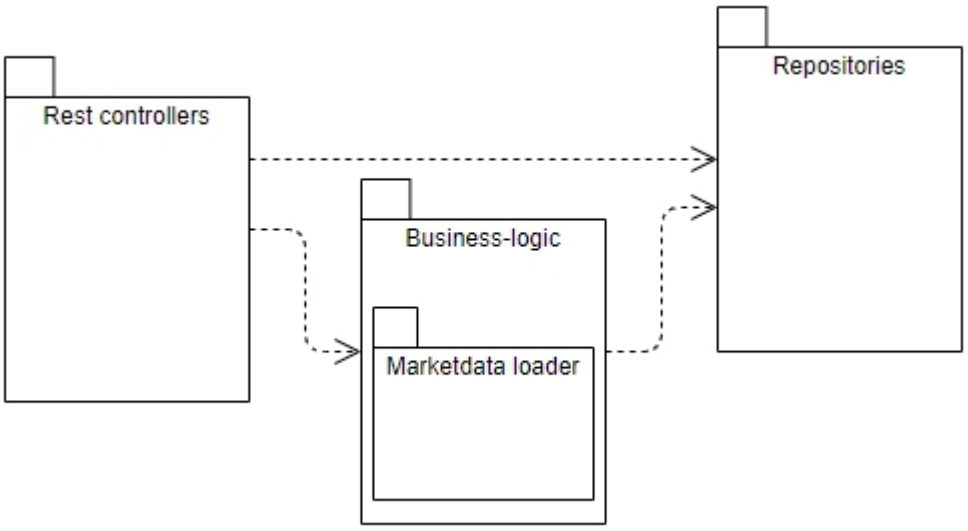

Рис. 6: Архитектура marketdata provider

Рассмотрим архитектуру модуля marketdata loader (рис. 7), отвечающего за загрузку данных о котировках активов из сторонних источников.

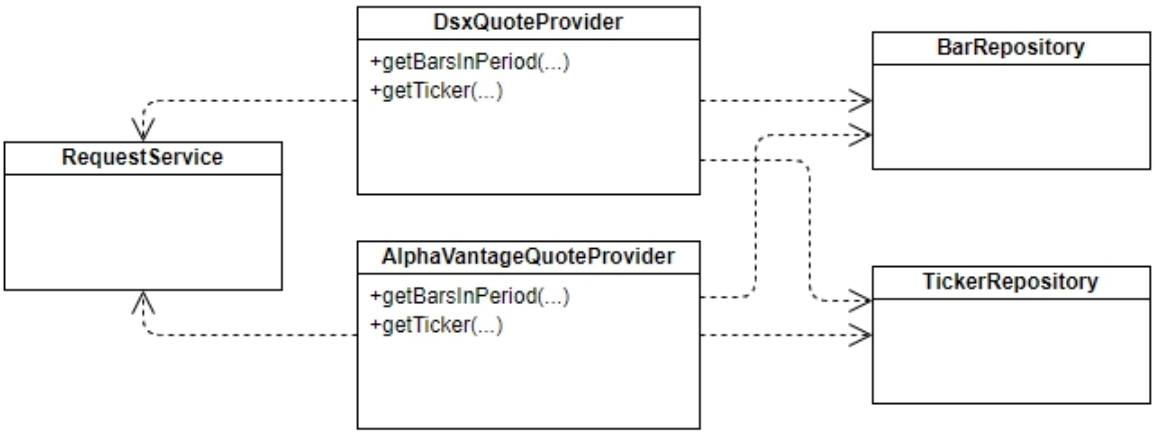

Рис. 7: Архитектура marketdata loader

Сервисы DsxQuoteProvider и AlphaVantageQuoteProvider отвечают за преобразование данных о котировках активов из форматов, предоставляемых dsxglobal.com [2] и alphavantage.co [1] соответственно, в фор-

мат проекта. “Bar” в данном контексте — отношение стоимости выбранного актива к доллару в определенный момент времени. Ticker — текущий курс выбранного актива по отношению к доллару. Провайдер загружает данные о котировках используя инструменты класса RequestService, после чего преобразует их в формат проекта и сохраняет в соответствующий репозиторий.

5. Реализация серверной части продукта

Рассмотрим технологии, используемые в данном проекте.

5.1. Технологии, используемые в данном проекте

- Язык программирования Java
  Для разработки слоя бизнес-логики современных приложений популярны языки программирования C++, C#, Java, Kotlin, Python, JavaScript и PHP. На языке C++ сложнее писать код, чем на остальных в этом списке, он сильно проигрывает в скорости разработки, при этом не сильно выигрывает в производительности. Из оставшихся я выделил языки C#, Java, Kotlin, так как для меня важны следующие характеристики:

  – Статическая типизация
    Статическая типизация позволяет отлавливать значительное количество ошибок на этапе компиляции.

  – Баланс между скоростью разработки и производительностью

  Из оставшихся был выбран язык Java, т.к. разработка на нём мало отличается от разработки на C# или Kotlin, при этом у меня был опыт работы с ним.

- Фреймворк Spring
  Были рассмотрены следующие фреймворки для реализации REST API:

  – Spring

  – ActFramework

  – Light-rest-4j

  – Ninja Web Framework

  – Play Framework

  – RESTEasy

  - Restlet
  - Spark Framework

  Все перечисленные инструменты позволяют легко создать работающую версию приложения за короткое время, однако у каждого из них есть ряд недостатков. Был выбран фреймворк Spring, так как это один из самых популярных java фреймворков для реализации REST API (Согласно исследованию JRebel labs [8]), следовательно у него огромное сообщество разработчиков, готовых ответить на все возникающие вопросы. Также Spring имеет подробную документацию с примерами, что встречается не у всех фреймворков из этого списка.

  Были использованы следующие компоненты фреймворка Spring:

  - Spring MVC
  - Spring Security

    В дополнение к Spring Security используется библиотека jjwt, которая реализует стандарт JSON Web Token [4]. JWT был выбран как следование требованию масштабируемости, так как он позволяет разбить систему на web-сервисы, не изменяя модуль, отвечающий за регистрацию и авторизацию пользователей. Также он позволяет создать несколько экземпляров приложения, запросы между которыми будут распределяться с помощью балансировщика нагрузки, не хранящего состояние (в отличие от аутентификации с использованием cookie).
  - Spring Data

- Фреймворк Swagger

  Swagger используется как следование требованию о документации API. Он позволяет решить довольно трудоемкую задачу — поддержание документации в актуальном состоянии. Все, что нужно сделать разработчику — интегрировать swagger в проект и время от времени корректировать автоматически генерируемую до-

кументацию. Причём изменять эту документацию можно как редактируя сгенерированный файл в формате YAML, так и с помощью аннотаций непосредственно в коде программы. Второй способ быстрее и удобнее для разработчика, однако у него есть свой недостаток. Со временем в некоторых файлах проекта количество текста, предназначенного для описания документации, может превысить количество кода, что негативно скажется на читаемости этих файлов. Однако этот недостаток проявляется скорее при командной разработке, поэтому в данном проекте я пишу документацию с помощью аннотаций непосредственно в коде программы.

- СУБД MySQL
  Для хранения данных о пользователях и их финансовых операциях в сервисе portfolio service я решил использовать СУБД MySQL по следующим причинам:
  - MySQL — реляционная БД, которая хорошо подходит для потенциально меняющихся данных
  - MySQL отлично интегрируется с фреймворком Spring
    Руководство по подключению и использованию базы данных в Spring описано на примере СУБД MySQL.
  - MySQL — это проверенная технология, которая используется крупными компаниями более 15 лет
  - Нет каких-либо требований на минимальный поддерживаемый объем данных

- СУБД Clickhouse
  Для хранения котировок активов в сервисе marketdata provider я решил использовать Clickhouse — колоночную СУБД от компании Яндекс, предназначенную для хранения большого количества статистических данных, так как необходимо эффективно решать следующие задачи:
  - Хранение большого количества не требующих операций изме-

нения и удаления данных, которые загружаются в базу кортежами большого размера с высокой частотой. Размер кортежа, после того как исторические данные о котировках активов успешно загружены, равен количеству поддерживаемых активов.

– В котировках требуется эффективно (потому что некоторые запросы являются блокирующими с точки зрения пользовательского интерфейса) выполнять запросы по подмножеству столбцов. Пример запроса по подмножеству столбцов: если мы хотим получить ежедневные котировки по какому-либо инструменту, нам нужно для каждого инструмента получить только один столбец, хранящий цену актива на момент закрытия торгов, и столбец с датой.

Яндекс позиционирует Clickhouse как СУБД, эффективно решающую эти задачи. Clickhouse успешно используется в сервисе Яндекс.Метрика, где по данным от компании Яндекс обрабатываются от сотен миллионов до более миллиарда строк и десятки гигабайт данных на один сервер в секунду.

5.2. Особенности реализации

Рассмотрим особенность реализации модуля marketdata loader. Упорядоченную пару активов (обмениваемый и запрашиваемый активы в рамках одной сделки) будем называть инструментом. Проблема заключается в том, что не все ресурсы, предоставляющие данные о котировках активов, умеют работать со всеми инструментами, построенными на базе активов, поддерживаемых этими ресурсами. Например, Dsx public api [2] предоставляет котировки по инструментам rub-eur (рубль-евро) и usd-rub (доллар-рубль), но не предоставляет котировки по инструменту usd-eur (доллар-евро). Есть два способа решения этой проблемы. Первый способ — прописать формулы, по которым можно получить данные по неподдерживаемым инструментам, непосредственно в коде программы. Таким образом, для того чтобы получить котировки по инструменту usd-eur в нашем примере, надо перемножить котировки по поддерживаемым инструментам usd-rub и rub-eur. Второй способ — использовать граф (Рис. 8), построенный по следующему принципу:

- Вершины графа — все поддерживаемые данным ресурсом активы
- Ребра графа — все поддерживаемые данным ресурсом инструменты

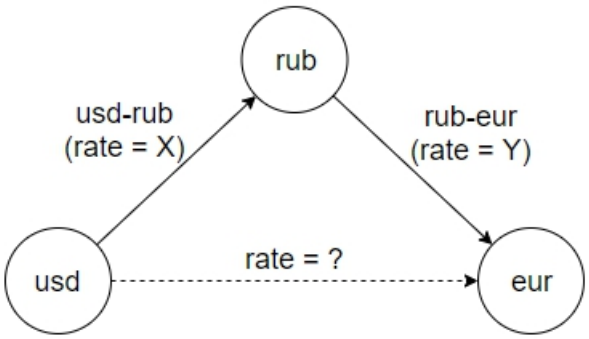

Рис. 8: Граф активов и инструментов

Для того чтобы получить котировки по инструменту usd-eur в нашем примере, надо найти кратчайший путь в графе (Рис. 8) из вершины usd в вершину eur, после чего перемножить котировки по всем

инструментам, встречающимся на пути. То есть, если в какой-то момент времени рубль обменивают на евро по соотношению X, доллар на рубль по соотношению Y, значит доллар обменивают на евро по соотношению X * Y. Первый способ проще и быстрее в реализации, когда нужны котировки по небольшому количеству активов (до 10 штук), однако второй способ более универсальный, он проще в реализации при большом количестве активов, а также позволяет заранее решить эту проблему для других ресурсов, предоставляющих котировки, так как граф достаточно реализовать только один раз. Dsx public api предоставляет статистику по 15 валютам и криптовалютам, поэтому был выбран второй способ. Графовые алгоритмы были реализованы с помощью библиотеки jgrapht-core.

Так как для получения котировок по какому-либо инструменту допускается перемножение котировок по другим инструментам (погрешность вычислений в таком случае пренебрежительно мала), было принято решение хранить только котировки активов по отношению к доллару. В таком случае котировки по любому инструменту (актив1-актив2) можно получить по формуле актив1-usd * usd-актив2. Такой подход позволяет сократить количество требуемой памяти с O(n^2) до O(n), где n — количество инструментов.

6. Развертывание серверной части продукта и её тестирование

Для того чтобы упростить интеграцию с приложениями-клиентами, было принято решение с начала разработки настроить тестовый сервер, где будет работать последняя версия серверной части приложения. Для упрощения интеграции изменений кода серверной части приложения и для упрощения процесса её развертывания были настроены процессы CI/CD. Их описание изображено на рис. 9.

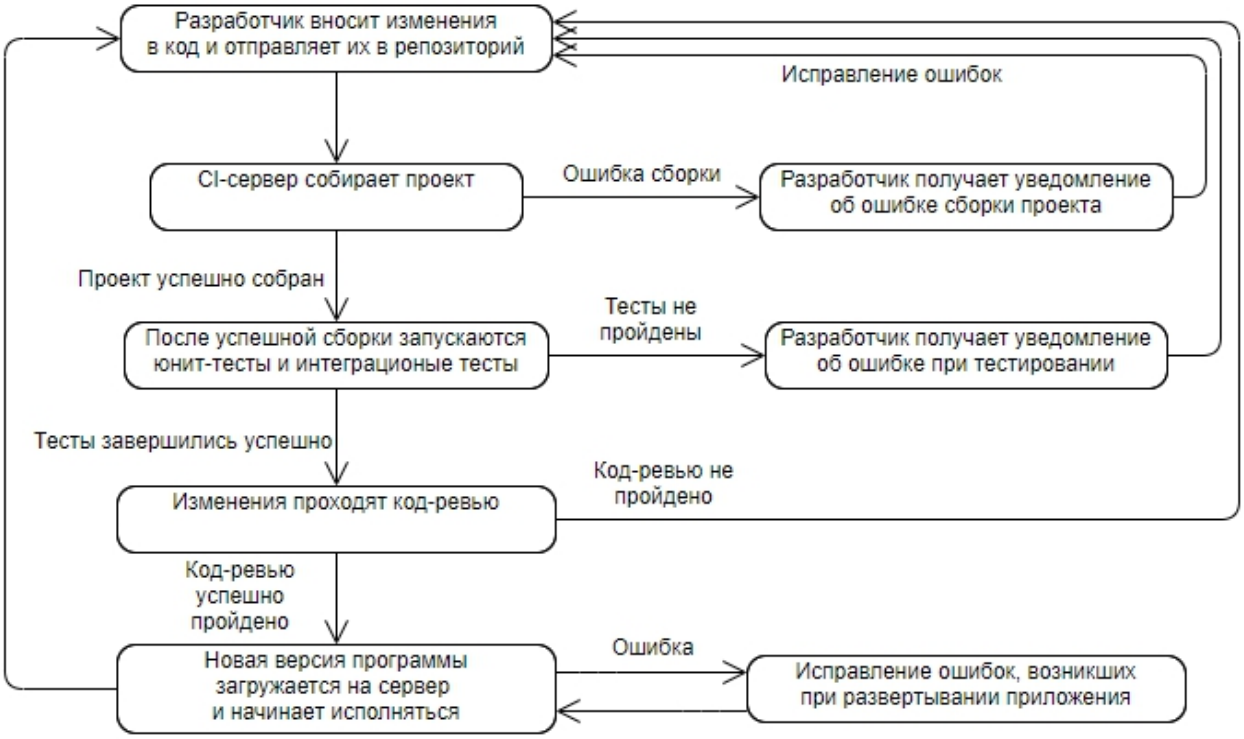

Рис. 9: Описание процессов CI/CD

Сборка проекта, а также тестирование и развертывание приложения осуществляются с использованием сервиса GitHub Actions. Для выявления ошибок до стадии развертывания код приложения покрывается юнит-тестами и интеграционными тестами. Для юнит-тестирования используются фреймворки JUnit и Mockito. JUnit — стандартный фреймворк для unit-тестирования Java-приложений. Mockito используется для тестирования с помощью mock-объектов. Для интеграционного тести-

рования используются фреймворк JUnit и компонент “Spring boot starter test” фреймворка Spring. Автоматические тесты покрывают 94% кода.

Заключение

В ходе данной работы были получены следующие результаты.

- Проанализированы существующие приложения для портфельного менеджмента
- Выявлены требования для серверной части продукта
- Спроектирована серверная часть продукта
- Реализована серверная часть продукта
- Серверная часть продукта развернута на сервере AWS, протестирована и успешно интегрирована с приложениями-клиентами

В дальнейшем планируется добавить интеграцию с бóльшим количеством торговых площадок, увеличить количество поддерживаемых сервисов, предоставляющих данные о котировках, а также увеличить количество отображаемых метрик по инвестиционному портфелю.

Список литературы

[1] Inc. Alpha Vantage. Alpha Vantage API Documentation. — 2020. — URL: dsxglobal.com/developers/publicApi (online; accessed: 01.05.2020).

[2] Inc. DSX. DSX Public API Documentation. — 2020. — URL: dsxglobal.com/developers/publicApi (online; accessed: 01.05.2020).

[3] Intelinvest. Как вести расчет основных показателей портфеля: подробный ликбез // blog.intelinvest.ru. — 2017. — URL: https://blog.intelinvest.ru/calculations-explained/ (дата обращения: 01.05.2020).

[4] Jones Michael B., Bradley John, Sakimura Nat. JSON Web Token (JWT) // RFC. — 2015. — URL: https://tools.ietf.org/html/rfc7519 (online; accessed: 01.05.2020).

[5] blockfolio.com. Blockfolio // blockfolio.com. — 2019. — URL: https://blockfolio.com (online; accessed: 13.12.2019).

[6] criteo. Мобильная коммерция вытесняет персональный компьютер // www.criteo.com. — 2017. — URL: https://www.criteo.com/ru/news/press-releases/2017/03/%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F-%D0%BA%D0%BE%D0%BC%D0%BC%D0%B5%D1%80%D1%86%D0%B8%D1%8F-%D0%B2%D1%8B%D1%82%D0%B5%D1%81%D0%BD%D1%8F%D0%B5%D1%82-%D0%BF%D0%B5%D1%80%D1%81%D0%BE/ (дата обращения: 01.05.2020).

[7] intelinvest.ru. Intelinvest // intelinvest.ru. — 2019. — URL: https://intelinvest.ru (online; accessed: 13.12.2019).

[8] labs JRebel. Java Web Frameworks Index // www.jrebel.com. — 2017. — URL: https://www.jrebel.com/blog/java-web-frameworks-index (дата обращения: 01.05.2020).

[9] www.investmentaccountmanager.com. Investment Account Manager // www.investmentaccountmanager.com. — 2019. — URL: https://www.investmentaccountmanager.com (online; accessed: 13.12.2019).

[10] www.personalcapital.com. Personal Capital // www.personalcapital.com. — 2019. — URL: https://www.personalcapital.com (online; accessed: 13.12.2019).

[11] Бахарев Игорь. Мобильные приложения в 2018 году: аналитика Criteo // e-pepper.ru. — 2018. — URL: https://e-pepper.ru/news/mobilnye-prilozheniya-v-2018-godu-analitika-criteo.html (дата обращения: 01.05.2020).

[12] Коцюба И.Ю., А.В. Чунаев, Шиков А.Н. Основы проектирования информационных систем. — Санкт-Петербург, Кронверкский пр., 49 : Университет ИТМО, 2015. — С. 202.