МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	Вычислительная техника

Направление подготовки 09.04.01 «Информатика и вычислительная техника»

Магистерская программа Вычислительные машины, комплексы, системы и сети

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

на тему

ИНТЕРАКТИВНАЯ СИСТЕМА КОЛЛЕКТИВНОГО УПРАВЛЕНИЯ ИНВЕСТИЦИОННЫМ ПАКЕТОМ

Студент _____ (подпись, дата) Краснов Андрей Андреевич (ФИО полностью)

Руководитель _____ (подпись, дата) Дубравин А. В. (фамилия, инициалы)

Нормоконтролёр 9.06.18 (подпись, дата) Кучин А. В. (фамилия, инициалы)

Рецензент нач. 2 отдела 2 СИ ООО НПП "Криптософт" (должность, место работы) _____ 13.06.18 (подпись, дата) Афонин А. Ю. (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 25.05.2018г. № 38-09-12)

Заведующий кафедрой _____ (подпись) Пащенко Д. В. (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 14.06.18г. № 5

Секретарь ГЭК _____ (подпись) Гурин Е. И. (фамилия, инициалы)

Пенза, 2018

Реферат

Пояснительная записка содержит 143 листа, 24 рисунка, 14 источников, 2 приложения.

ФОНДОВЫЙ РЫНОК, БИЗНЕС-АНАЛИТИКА, ИНВЕСТОРЫ, ЭКСПЕРТЫ, ВЕБ-СЕРВИС, ПРОГНОЗИРОВАНИЕ, КОЛЛЕКТИВНОЕ УПРАВЛЕНИЕ.

Цель работы: выполнить анализ предметной области – фондового рынка и процесса влияния на него современных информационных технологий. Провести исследование существующих методологий проектирования бизнес-процессов и последующее проектирование основных процессов системы в соответствии с выбранной методологией. Сформировать требования к программно-аппаратному комплексу организационных, информационных и технических решений. Разработать архитектуру системы. Определить оптимальную методологию разработки ПО. Выполнить программную реализацию на основе сформированных ранее данных и дальнейшее тестирование системы.

В результате проведенной работы была разработана *интерактивная система коллективного управления инвестиционным пакетом*, реализующая платформу для взаимовыгодного сотрудничества клиентов – лиц, заинтересованных в приобретении бизнес-аналитики по тому или иному финансовому инструменту на определенный период, и экспертов системы – пользователей, обладающих информацией, доступной узкому кругу лиц, особыми навыками или талантами в области анализа текущего состояния и прогнозирования тенденций изменения тех или иных финансовых инструментов фондового рынка.

Разработанная система представляет собой демонстрационный образец, подразумевающий доработку и дальнейшее развитие для развёртывания в сети Интернет и практического использования.

СОДЕРЖАНИЕ

Введение **4**

1 ОПИСАНИЕ СИСТЕМЫ **12**

1.1 БИЗНЕС-ПРОЦЕСС ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ С ТОЧКИ ЗРЕНИЯ КЛИЕНТА-ИНВЕСТОРА 15

1.2 БИЗНЕС-ПРОЦЕСС ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ С ТОЧКИ ЗРЕНИЯ КЛИЕНТА-ЭКСПЕРТА 20

1.3 ВЫВОДЫ 24

2 АРХИТЕКТУРА ПРИЛОЖЕНИЯ **25**

2.1 ТРЕБОВАНИЯ К ПРОГРАММНО-АППАРАТНОМУ КОМПЛЕКСУ ОРГАНИЗАЦИОННЫХ, ИНФОРМАЦИОННЫХ И ТЕХНИЧЕСКИХ РЕШЕНИЙ 25

2.2 АРХИТЕКТУРА СИСТЕМЫ 25

4 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ **33**

4.1 ОБОСНОВАНИЕ ВЫБОРА СРЕДСТВ И ТЕХНОЛОГИИ ДЛЯ РЕАЛИЗАЦИИ ПОСТАВЛЕННОЙ ЗАДАЧИ 33

4.2 ВЫБОР МЕТОДОЛОГИИ РАЗРАБОТКИ 34

4.3 ОПИСАНИЕ МОДУЛЕЙ И ФУНКЦИЙ 37

5 ТЕСТИРОВАНИЕ **43**

6 НАПРАВЛЕНИЯ РАЗВИТИЯ **59**

6.1 РАЗРАБОТКА И ВНЕДРЕНИЕ МЕХАНИЗМА «КОЛЛЕКТИВНОГО УПРАВЛЕНИЯ» 59

6.2 ИССЛЕДОВАНИЕ И ДОРАБОТКА СИСТЕМЫ С ТОЧКИ ЗРЕНИЯ ВОПРОСА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ 60

6.3 РЕАЛИЗАЦИЯ КОНЦЕПЦИИ «FRONT END – BACK END» 61

6.4 РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ 61

6.5 РЕГИСТРАЦИЯ ТОВАРНОГО ЗНАКА 62

7 ЗАКЛЮЧЕНИЕ **63**

8 СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ **65**

Приложение А. Текст программы **67**

Приложение Б. Презентация Microsoft Power Point **130**

Введение

Ры́нок це́нных бума́г, фо́ндовый ры́нок (англ. stock market, англ. equity market) — совокупность экономических отношений по поводу выпуска и обращения ценных бумаг между его участниками. История рынка ценных бумаг насчитывает несколько веков. Его возникновение часто связывают с созданием в XV—XVI веках рынка **государственных** ценных бумаг. В этот период государства для привлечения дополнительных денежных средств, необходимых для покрытия дефицита денежных средств на государственные нужды, стали выпускать и размещать ценные бумаги как внутри страны, так и за рубежом.

С момента возникновения и на протяжении всего исторического периода данная область экономики развивалась и эволюционировала вместе с технологическим прогрессом и общим уровнем развития человеческого общества. В развитии биржевой торговли ценными бумагами наблюдались одинаковые тенденции, характерные для большинства развитых стран. Вследствие этого можно говорить о фондовом рынке, как об общечеловеческом понятии, минуя незначительные региональные особенности.

В XX веке фондовый рынок продолжал развиваться не только количественно, но и качественно, благодаря появлению новых технологий. Для второй половины XX века характерно появление организованного внебиржевого рынка, основанного на компьютерных технологиях.

Начало XXI века ознаменовалось очередным витком эволюции фондового рынка на фоне развития информационных технологий и грандиозного скачка доступности их для широких масс населения. В рамках этого процесса возникают совершенно новые понятия, одним из которых является «алгоритмическая торговля».

Алгоритмическая торговля с самого своего появления в начале 90-х годов прошлого века была инструментом крупных инвесторов. Децимализация (переход на Нью-Йоркской бирже к использованию в торговле акциями на десятичную систему — минимальный шаг цены стал равняться 1 центу, а не 1/16 доллара), технологии прямого доступа на рынок (Direct Market Access, DMA), 100% электронные биржи, снижение комиссий бирж и брокеров, появление различных биржевых площадок в США и в других странах — все это привело к взрывному росту числа трейдеров, использующих алгоритмы.

Профессор математики Нью-Йоркского Университета и эксперт по финансовым рынкам Марко Авелланеда (Marco Avellaneda) описывает цели использования алгоритмов в биржевой торговле следующим образом. По мнению профессора, в случае крупных институциональных инвесторов они применяются главным образом не для максимизации возможной прибыли с конкретной сделки, а для контроля рыночного риска и издержек исполнения ордера.

Проще говоря, обычно крупным инвесторам нужно совершать операции с большим объёмом акций. Часто объём сделки выше, чем рынок может «переварить» без изменения цены акции. Необходимость совершить покупку огромного количества акций приведет к изменению их цены и появлению так называемого «проскальзывания». Таким образом, исполнить весь приказ по одной цене не удастся — сначала сделки будут проходить по нужной цене, но постепенно она будет становиться все менее выгодной.

Чтобы этого избежать, необходимо разбивать крупные ордера на более мелкие, которые исполняются через интернет в течение минут, часов или дней. Чтобы сделать это максимально выгодно, алгоритм должен контролировать среднюю стоимость акции. Оценить ее можно сравнив с

рыночным «бенчмарком» — глобальной средней ценой за день, ценой закрытия или открытия и т.п.

Но проблема определения того, как именно разбивать крупный приказ на более мелкие, является не единственной. Алгоритм также должен решить, как именно выводить ордер на рынок — в виде лимитного или рыночного приказа — и по какой цене. Необходимо добиться наилучшей цены для каждого такого дочернего приказа.

Развитие финансовых рынков и появление новых торговых инструментов сделали эту задачу куда более сложной и интересной. Времена, когда клиенты могли передать заявки своим брокерам только по телефону или факсу, ушли в прошлое.

Сейчас существуют разные способы подключения к электронным торгам (рисунок 1). Например, существует возможность подключения торгового робота к брокерской системе с помощью API — в таком случае приказы отправляются в брокерскую систему, а оттуда попадают на биржу (например, у брокерской компании ITinvest есть свой API-интерфейс SmartCOM).

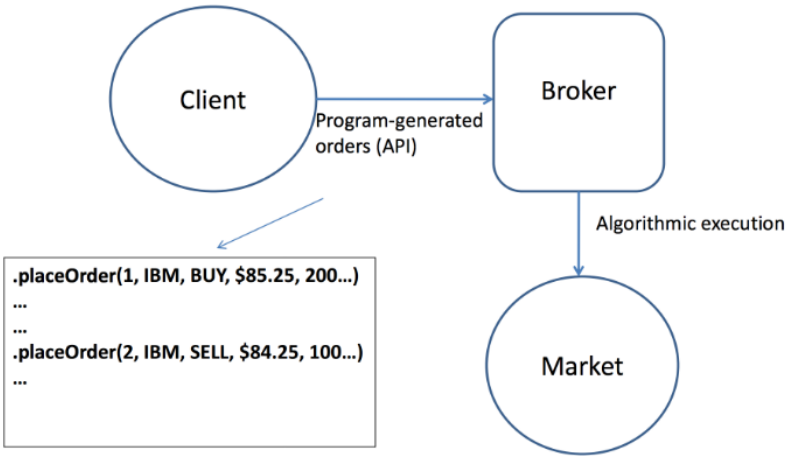

Рисунок 1 – схема подключения торгового бота к брокерской системе с помощью API

В случае алгоритмической торговли, как правило, важна скорость работы стратегии, поэтому многие трейдеры предпочитают использовать технологию прямого доступа на рынок (direct market access, DMA — ITinvest предоставляет такой доступ к российским и зарубежным биржам). В случае ее применения торговый робот взаимодействует напрямую с торговой системой биржи, минуя систему брокера, что позволяет выиграть время (рисунок 2).

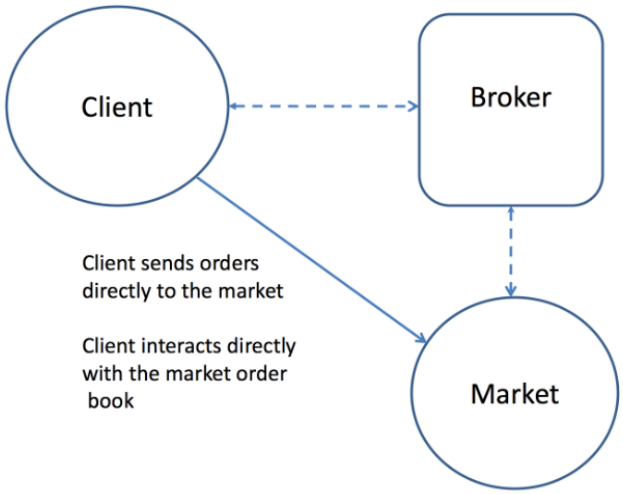

Рисунок 2 – схема подключения торгового бота для прямого доступа на рынок (direct market access, DMA)

Но это далеко не самый сложный вариант торговли. Появление большого количества различных торговых площадок привело к развитию алгоритмов «умной маршрутизации» приказов — такие системы не только пытаются совершать самые выгодные сделки на конкретной бирже, но еще и анализируют, на какой из доступных площадок в настоящий момент условия лучше, чтобы направить приказ именно туда.

Таким образом, существует три направления развития современных алгоритмов в рамках исследуемой предметной области:

- Алгоритмы макротрейдинга — определяют торговую стратегию;
- Алгоритмы микротрейдинга — собственно, торговые «движки» выставления ордеров;
- Алгоритмы умной маршрутизации — в случае, если работа ведется на нескольких биржах одновременно.

Помимо «экзекьюшн-стратегий», существует и целый ряд стратегий, направленных на извлечение прибыли с помощью других моделей.

Некоторые из них представлены следующими:

- Арбитражные стратегии — подмножество стратегий парного трейдинга, которые основаны на анализе соотношений цен двух высоко коррелированных между собой финансовых инструментов. В случае арбитража, такая пара состоит из одинаковых или связанных активов, корреляция которых близка к единице — например, акций одной и той же компании на разных биржах. Для успешной торговли в рамках арбитражных стратегий критически важна скорость получения данных и выставления/изменения заявок на покупку или продажу.
- Предоставление ликвидности (маркет-мейкинг) — маркетмейкинг предполагает поддержание спредов на покупку и продажу финансового инструмента. Маркетмейкеры являются основными поставщиками моментальной ликвидности, поэтому биржи часто привлекают их к работе с неликвидными инструментами с помощью предоставления льготных условий.
- Предсказание цены — стратегии, которые анализируют различные данные (в том числе с помощью индикаторов технического анализа) для построения гипотез о том, в какую сторону может

двинуться цена финансового инструмента в заданный промежуток времени.

В рамках высокочастотной торговли, для того, чтобы «предсказать» движение цены, алгоритм должен смоделировать скрытую ликвидность рынка при данной ликвидности заявок на покупку и продажу. «Истощение» очереди заявок на покупку или продажу может свидетельствовать о скором движении цены (рисунок 3).

Рисунок 3 – очереди заявок

Изменение цены возникает, когда на одном из уровней цены исчезают все заявки на покупку или продажу, и существует следующий уровень цен бид и аск. Вероятность того, что очередь заявок аск истощится ранее, чем очередь заявок бид, высчитывается следующим образом:

$$u(x,y)=\frac{1}{2}\left(1-\frac{\tan^{-1}\left(\sqrt{\frac{1+\rho}{1-\rho}}\,\frac{y-x}{x+y}\right)}{\tan^{-1}\left(\sqrt{\frac{1+\rho}{1-\rho}}\right)}\right)$$

$$\rho=0 \quad \Rightarrow \quad u(x,y)=\frac{2}{\pi}\tan^{-1}\left(\frac{x}{y}\right)$$

$$\rho=-1 \quad \Rightarrow \quad u(x,y)=\frac{x}{x+y}$$

Итоговая формула вероятности повышения цены:

$$p\uparrow(x,y,H)=u(x+H,y+H)$$

, где H — скрытая ликвидность рынка, то есть сделки, которые неизвестны широкой общественности (например, сделки крупных финансовых организаций, которые заключаются за пределами бирж).

Процедура оценки выглядит следующим образом:

- На первом этапе собранные данные разделяются по биржам, за один раз анализируется один торговый день;
- Котировки значений бид и аск компонуются по децилям. Для каждого такого набора (i,j) вычисляется частота повышения цены u_ij.
- Подсчитывается число появлений каждой величины d_ij.
- Производится анализ соответствия модели с помощью метода наименьших квадратов:

$$\min_{H,\rho} \sum_{ij=1}^{10} d_{ij} \left[u_{ij} - \frac{1}{2} \left(1 - \frac{\tan^{-1}\left(\sqrt{\frac{1+\rho}{1-\rho}} \frac{j-i}{j+i+2H} \right)}{\tan^{-1}\left(\sqrt{\frac{1+\rho}{1-\rho}} \right)} \right) \right]^2$$

Таким образом, на многих фондовых площадках (например, в США и России) оборот алгоритмической торговли уже довольно давно составляет более 50%. При этом часто алгоритмы используются не только для того, чтобы «опередить» конкурентов в скорости совершения транзакций и заработать на этом.

Крупные игроки могут применять этот инструмент для того, чтобы разбивать крупные сделки на более мелкие, которые позволяют осуществить операцию с заданным количеством финансового инструмента, не сдвигая его рыночную цену в ту или иную сторону. Для этого используются алгоритмы TWAP, VWAP и PoV.

Кроме того, алгоритмы используются для реализации «квантовых стратегий», таких как арбитраж или маркетмейкинг. Помимо этого, существуют возможности по подсчету вероятности изменения цены конкретных финансовых инструментов.

Однако, на данный момент механизмы машинного прогнозирования цен показывают свою эффективность в рамках такого сегмента фондового рынка, как высокочастотная торговля, но в сегменте долгосрочного прогнозирования – макротрейдинга – по-прежнему проигрывают специалистам в этой области и инсайдерам.

Данный факт свидетельствует об актуальности создания платформы, разрабатываемой в рамках магистерской диссертации «Интерактивная система коллективного управления инвестиционным пакетом».

1 ОПИСАНИЕ СИСТЕМЫ

В процессе выполнения научно-исследовательской работы реализована интерактивная система коллективного управления инвестиционным пакетом.

Система представляет собой электронную торговую площадку предоставления услуг бизнес-аналитики в сегменте фондового рынка. Программно-аппаратный комплекс организационных, информационных и технических решений, обеспечивает взаимодействие бизнес-аналитиков (экспертов), клиентов (инвесторов) и бирж.

В рамках системы коллективного управления инвестиционным пакетом реализуется бизнес-процесс, субъектами которого являются:

- «Эксперты» (бизнес-аналитики);
- «Клиенты» (инвесторы);
- «Биржи»;

Эксперты – пользователи, обладающие информацией, доступной узкому кругу лиц, особыми навыками или талантами в области анализа текущего состояния и прогнозирования тенденций изменения тех или иных финансовых инструментов фондового рынка. Имеют цель, направленную на обогащение, путем удовлетворения спроса «Клиентов». При этом – ограждены от финансовых рисков, однако, риск-фактором является потеря рейтинга, являющегося основополагающей характеристикой эксперта в рамках системы коллективного управления инвестиционным пакетом, обеспечивающей дальнейший финансовый успех эксперта.

Клиенты – потенциальные инвесторы. Круг лиц, заинтересованный в приобретении бизнес-аналитики по тому или иному инструменту на определенный период. Имеют цель, направленную на обогащение, путем получения прибыли от операций с ценными бумагами \ валютами \ криптовалютами и т. д. при регистрации на бирже ордеров с

характеристиками (такими как: период, стоп лосс и тейк профит), указанными бизнес-аналитиком. Система коллективного управления инвестиционным пакетом предоставляет клиентам-инвесторам возможность получения прибыли, значительно превышающей прибыль по банковским вкладам, при этом минимизируя существующие риски потери инвестиций системой корреляции рейтинга экспертов – «цена» ошибки эксперта (потеря рейтинга) с более высокой репутацией значительно более высока – что позволяет исключить фактор случайности и говорить о профессионализме эксперта в данной области.

Биржи – юридическое лицо, обеспечивающее регулярное функционирование организованного рынка биржевых товаров, валют, ценных бумаг и производных финансовых инструментов, а также предоставляющее API для взаимодействия через сеть Интернет.

Бизнес-процесс – совокупность взаимосвязанных мероприятий или работ, направленных на создание определённого продукта или услуги для потребителей. Существует множество нотаций, применяемых для моделирования бизнес-процессов. Наиболее популярные – представлены следующими:

- **BPMN** — функциональная последовательность работ;
- **EPC** — событийная последовательность работ;
- **IDEF0** — логическая последовательность работ.

В результате анализа представленных выше нотаций, в качестве методологии для разработки модели бизнес-процесса системы была выбрана IDEF0**.**

Выбор IDEF0 обусловлен следующим фактором: в отличие от аналогов, данная методология позволяет рассматривать логические отношения между работами, а не их временную последовательность (поток

работ). Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов, что позволило построить максимально эффективную модель бизнес-процесса системы.

Функциональная модель IDEF0 – это набор блоков, каждый из которых представляет собой «черный ящик» с входами и выходами, управлением и механизмами, которые детализируются (декомпозируются) до необходимого уровня. Соединяются функции между собой при помощи стрелок и описаний функциональных блоков. При этом, каждый вид стрелки или активности имеет собственное значение. Данная модель позволяет описать все основные виды процессов, как административные, так и организационные.

Стрелки могут быть:

- Входящие – вводные, которые ставят определенную задачу.
- Исходящие – выводящие результат деятельности.
- Управляющие (сверху вниз) – регуляторы управления (положения, инструкции и пр).
- Механизмы (снизу вверх) – что используется для того, чтобы произвести необходимую работу.

Входящие и исходящие называются Input и Output соответственно.

Стрелки подписываются при помощи имен существительных (опыт, план, правила), а блоки – при помощи глаголов, т.е. в них описываются действия, которые производятся (создать товар, заключить договор, произвести отгрузку).

IDEF0 – это очень простой и одновременно наглядный язык описания бизнес-процессов. С помощью этого стандарта возможна передача информации между разработчиками, консультантами и пользователями. Стандарт очень тщательно разрабатывался, он удобен для проектирования,

универсален. Для работы с ним существует множество инструментов, например, VISIO, BPWIN, ERWIN, Bussines studio и т.д.

В рамках научно-исследовательской работы для создания моделей бизнес-процесса в качестве инструмента для работы с IDEF0 была выбрана Microsoft Visio 2010.

При создании функциональной модели ключевыми параметрами являются цель и точка зрения. Исходя из них, моделирование одних и тех же процессов может выглядеть по-разному.

1.1 БИЗНЕС-ПРОЦЕСС ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ С ТОЧКИ ЗРЕНИЯ КЛИЕНТА-ИНВЕСТОРА

Бизнес-процесс, реализуемый в рамках системы коллективного управления инвестиционным пакетом, с точки зрения клиента представлен следующей функциональной моделью верхнего уровня (рисунок 4):

Рисунок 4 – бизнес-процесс с точки зрения клиента-инвестора

«Инвестиции» (I1) – вводная, которая необходима для начала работы.

Управляющие регуляторы это:

- «Доступные финансовые средства» (C1) – объем денежных средств, которые клиент готов привлечь для инвестирования, а также оплаты услуг сервиса;
- «Ожидаемая прибыль» (C2) – минимальный объем денежных средств, которые клиент ожидает получить в результате инвестиции;
- «Возможные потери» (C3) – максимальный объем денежных средств, которые клиент готов потерять в результате инвестиции;

- «Период инвестирования» (С4) – период, на который клиент готов совершить инвестирование с соответствующим получением ожидаемого объема прибыли в конце периода.

В роли «Механизмов» выступают сервис (М1), клиент (М2), биржа (М3). Клиент совершает оплату услуг сервиса и дальнейшее инвестирование на основе приобретенной бизнес-аналитики. Сервис – предоставляет бизнес-аналитику по определенным критериям-регуляторам клиента. Биржа – платформа для инвестирования клиента.

«Выход» процесса (О1) представляет собой прибыль, которую получает клиент в результате завершения периода инвестирования по рекомендации сервиса на той или иной бирже.

Таки образом, заданы основные параметры процесса, его вход, выход, а также все необходимое для успешного выполнения. Однако, это – только основная схема работы бизнес-процесса. Для детализации – выполнена декомпозиция общего блока «Воспользоваться сервисом для получения максимальной прибыли» на связанные между собой блоки (рисунок 5):

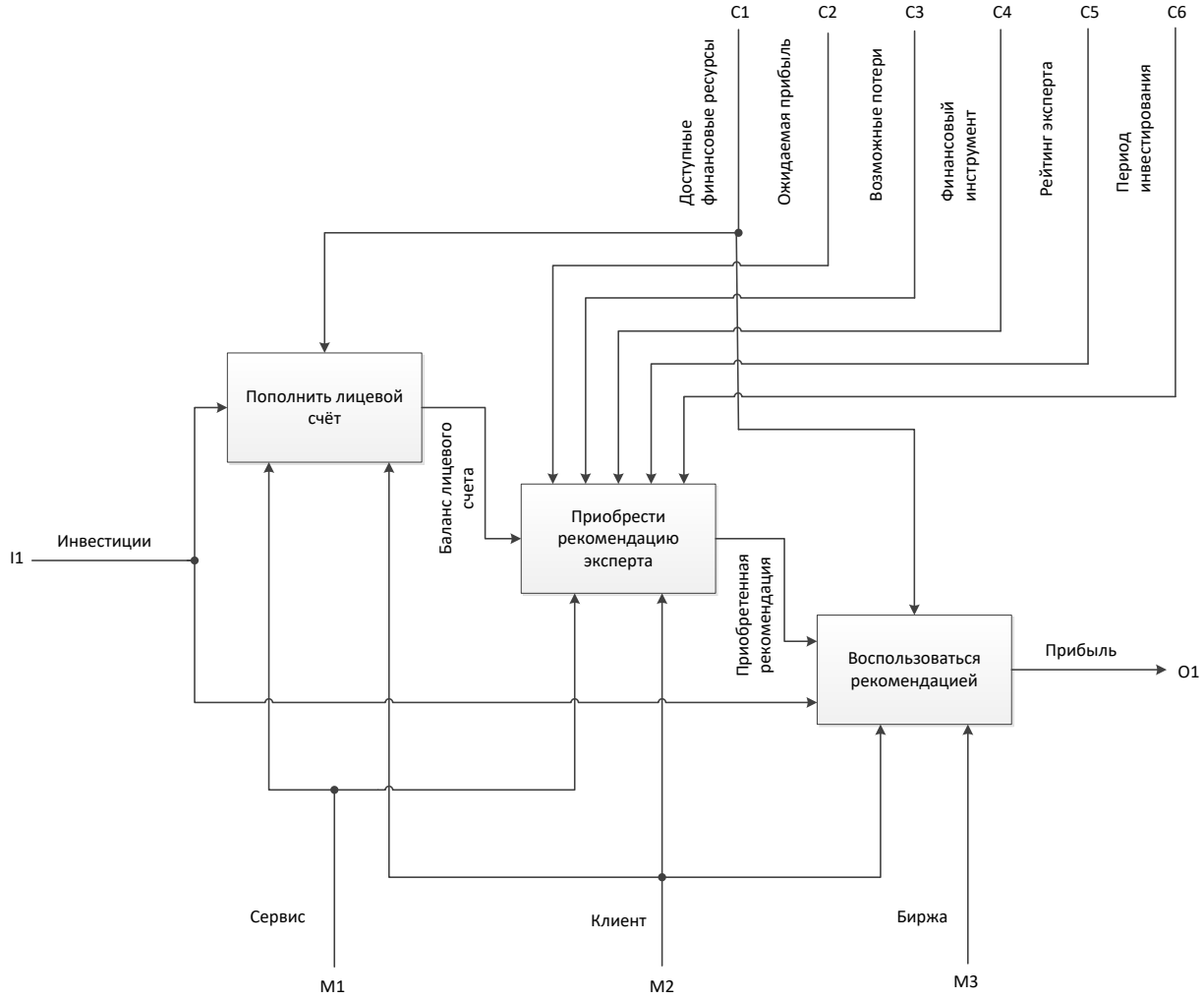

Рисунок 5 – детализированный бизнес-процесс с точки зрения клиента-инвестора

В результате декомпозиции основного блока бизнес-процесс (с точки зрения клиента-инвестора) делится на три основных этапа:

1. «Пополнить лицевой счёт» – пополнить свой лицевой счет или убедиться в достаточности средств в аккаунте сервиса (интерактивной системы коллективного управления инвестиционным пакетом);
2. «Приобрести рекомендацию эксперта»;
3. «Воспользоваться рекомендацией».

На схеме наглядно представлено, как и на каком этапе задействованы те или иные механизмы и управляющие элементы.

Так, клиент при пополнении счета должен внести некоторую сумму в оплату услуг сервиса, руководствуясь доступным объемом денежных средств.

Далее, клиент с помощью сервиса подбирает и покупает доступный (по текущему состоянию лицевого счета клиента) совет, сопоставляя свой выбор со следующими регуляторами, часть из которых была описаны выше при описании общего уровня детализации бизнес-процесса:

- «Доступные финансовые средства» (C1) – объем денежных средств, которые клиент готов привлечь для инвестирования, а также оплаты услуг сервиса;
- «Ожидаемая прибыль» (C2) – минимальный объем денежных средств, которые клиент ожидает получить в результате инвестиции;
- «Возможные потери» (C3) – максимальный объем денежных средств, которые клиент готов потерять в результате инвестиции;
- «Финансовый инструмент» (C4) – определенный финансовый документ (ценная бумага, фьючерс, криптовалюта и т. п.), в который будет проводиться инвестирование;
- «Рейтинг эксперта» (C5) – уровень рейтинга эксперта в рамках сервиса, который соответствует уровню доверия клиента;
- «Период инвестирования» (C6) – период, на который клиент готов совершить инвестирование с соответствующим получением ожидаемого объема прибыли в конце периода.

На третьем этапе «Воспользоваться рекомендацией» клиент инвестирует некоторый объем денежных средств, в соответствии с приобретенной рекомендацией эксперта, на той или иной бирже. Главным (и

единственным) регулятором на этом этапе являются имеющиеся в распоряжении клиента-инвестора финансовые ресурсы.

В результате успешного завершения бизнес-процесса формируется прибыль, которую получает клиент в результате завершения периода инвестирования по рекомендации сервиса на той или иной бирже.

1.2 БИЗНЕС-ПРОЦЕСС ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ С ТОЧКИ ЗРЕНИЯ КЛИЕНТА-ЭКСПЕРТА

Бизнес-процесс, реализуемый в рамках системы коллективного управления инвестиционным пакетом, с точки зрения эксперта представлен следующей функциональной моделью верхнего уровня (рисунок 6):

Рисунок 6 – бизнес-процесс с точки зрения эксперта

«Опыт» (I1) или «Инсайдерская информация» (I2) – вводные, которые необходимы для начала работы – то, что позволит эксперту сформировать прогноз для рекомендации.

Управляющие регуляторы это:

- «Финансовый инструмент» (C1) – определенный финансовый документ (ценная бумага, фьючерс, криптовалюта и т. п.), состояние которого прогнозируется экспертом;
- «Период инвестирования» (C2) – период, на который эксперт готов сформировать рекомендацию.
- «Состояние финансового инструмента» (C3) – текущая стоимость финансового инструмента.

В роли «Механизмов» выступают эксперт (M1), сервис (M2), клиент (M3).

Эксперт выполняет формирование рекомендации. Далее – в течение периода прогноза – эксперт ожидает, что клиенты приобретут его бизнес-аналитику. Сервис периодически сверяет зарегистрированную в системе рекомендацию эксперта с текущим состоянием финансового инструмента (относительно той или иной биржи), а затем – по достижению завершения периода прогноза, или при достижении тейк профита или стоп лосса – сервис анализирует точность соответствия прогноза эксперта реальному положению вещей. По результатам проверки выполняется начисление (или списание – в случае ошибки) баллов рейтинга эксперта в системе.

«Выход» процесса (O1) представляет собой прибыль, которую получает эксперт в результате приобретения клиентами-инвесторами его рекомендации.

«Выход» процесса (О2) представляет собой начисление очков рейтинга эксперта.

Таки образом, заданы основные параметры процесса, его входы, выходы, а также все необходимое для успешного выполнения. Однако, это – только основная схема работы бизнес-процесса. Для детализации процесса выполнена декомпозиция общего блока «Воспользоваться сервисом для продажи рекомендации» на связанные между собой блоки (рисунок 7):

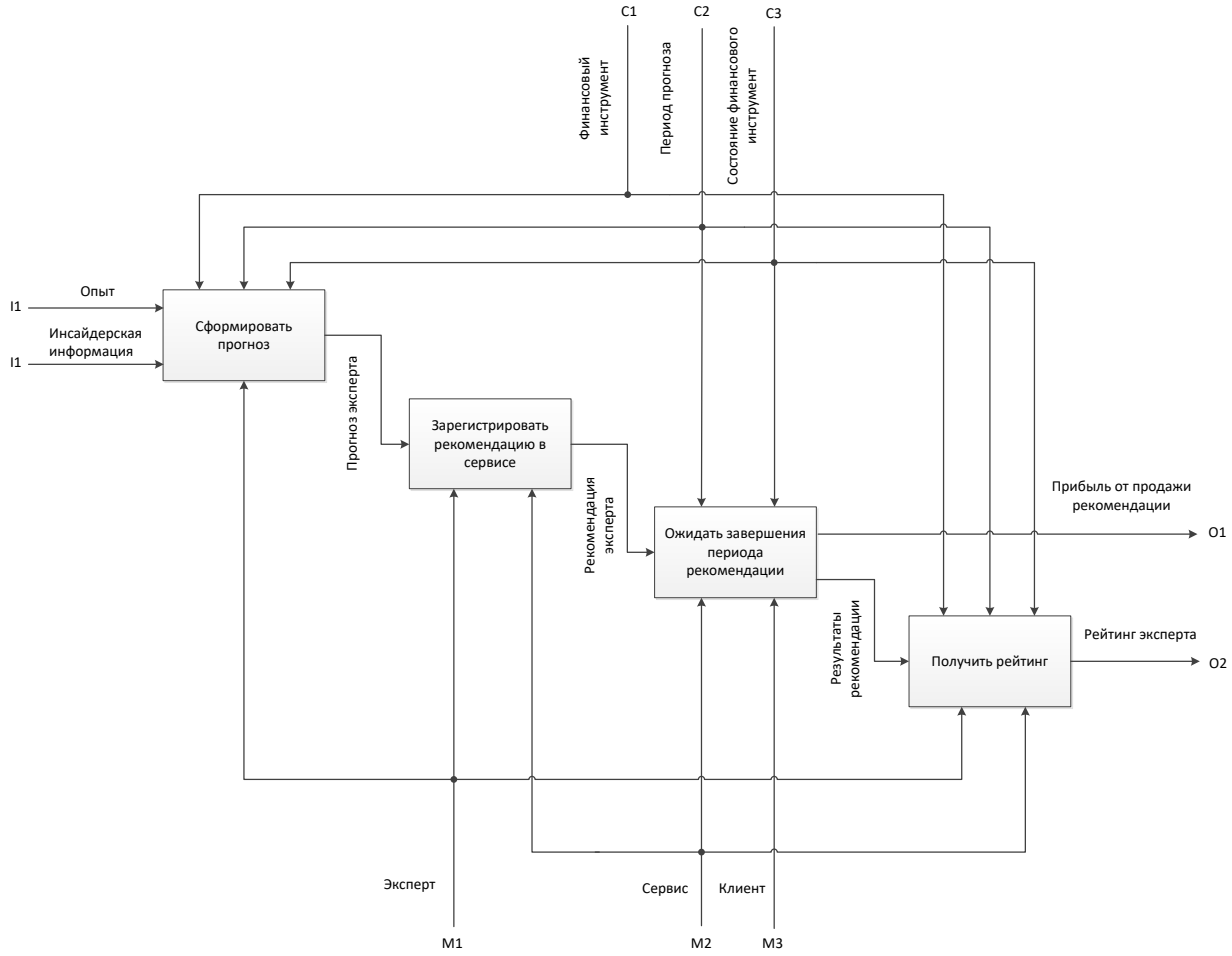

Рисунок 7 – детализированный бизнес-процесс с точки зрения эксперта

В результате декомпозиции основного блока бизнес-процесс (с точки зрения эксперта системы) делится на четыре основных этапа:

1. «Сформировать прогноз»;
2. «Зарегистрировать рекомендацию в сервисе»;

3. «Ожидать завершения периода рекомендации»;
4. «Получить рейтинг».

На схеме наглядно представлено, как и на каком этапе задействованы те или иные механизмы и управляющие элементы.

Так, эксперт формирует прогноз на основе своего профессионального и жизненного опыта или \ и инсайдерской информации, руководствуясь определенным финансовым инструментом, его текущим состоянием и планируемым периодом прогнозирования.

Далее, эксперт регистрирует в системе рекомендацию (с указанием основных характеристик: периода, тейк профита, стоп лоса).

На третьем этапе «Ожидать завершения периода рекомендации» клиент с помощью сервиса приобретает рекомендацию эксперта для инвестирования. В свою очередь, сервис периодически сверяет зарегистрированную в системе рекомендацию эксперта с текущим состоянием финансового инструмента на предмет достижения тейк профита или стоп лосса, а также достижения завершения периода прогноза. Регуляторы на этом этапе представлены текущим состоянием финансового инструмента и периодом прогноза. Одна из исходящих стрелок блока является выходом бизнес-процесса (О1) – «Прибыль от продажи рекомендации».

На четвертом этапе сервис проверяет точность соответствия прогноза эксперта реальному положению вещей. По результатам проверки выполняется начисление (или списание – в случае ошибки) баллов рейтинга эксперта в системе – выход «Рейтинг эксперта» (О2).

1.3 ВЫВОДЫ

Таким образом, интерактивная система коллективного управления инвестиционным пакетом, разработанная в процессе выполнения научно-исследовательской работы, обеспечивает заинтересованность всех субъектов бизнес-процесса во взаимовыгодном сотрудничестве.

Так, клиент получает возможность получения прибыли, значительно превышающей прибыль по банковским вкладам, при этом минимизируя существующие риски потери инвестиций системой корреляции рейтинга экспертов. Эксперты – получают прибыль, путем удовлетворения спроса «Клиентов» на бизнес-аналитику, и при этом – ограждены от финансовых рисков. А биржи, получающие финансовую прибыль от оборота, – выигрывают от привлечения новых клиентов в инвестирование.

2 АРХИТЕКТУРА ПРИЛОЖЕНИЯ

2.1 ТРЕБОВАНИЯ К ПРОГРАММНО-АППАРАТНОМУ КОМПЛЕКСУ ОРГАНИЗАЦИОННЫХ, ИНФОРМАЦИОННЫХ И ТЕХНИЧЕСКИХ РЕШЕНИЙ

На предыдущем этапе выполнения научно-исследовательской работы, направленной на разработку интерактивной системы коллективного управления инвестиционным пакетом, были сформированы функциональные модели бизнес-процессов системы, отражающие логические отношения между соподчиненными этапами работы сервиса. Однако, для проектирования архитектуры информационной системы, помимо реализации бизнес-логики внутренних процессов, необходимым условием является формирование требований к программно-аппаратному комплексу организационных, информационных и технических решений.

Программно-аппаратный комплекс организационных, информационных и технических решений должен обеспечивать отказоустойчивую работу интерактивной системы коллективного управления инвестиционным пакетом в течение длительного периода времени. Единственным основанием для приостановления функционирования сервиса является запланированная профилактика или техническое обслуживание системы с предварительным уведомлением всех пользователей системы.

Основой отказоустойчивой работы сервиса является система распределенного хранения данных, а также механизм репликации и балансировки нагрузки.

2.2 АРХИТЕКТУРА СИСТЕМЫ

На основе сформированных требований была разработана следующая архитектура системы (рисунок 8):

- «Веб-портал» – сервис, реализующий идентификацию и авторизацию пользователей, а также формирующий контент, отображаемый пользователю;
- «Сервис балансировки нагрузки» – сервис, реализующий логику распределения запросов от клиентов системы к сервисам хранения данных.
- «Сервис хранения данных» – сервис, непосредственно взаимодействующий с хранилищем данных, возвращающий их по запросам.
- «Сервис верификации рекомендаций» – сервис, выполняющий мониторинг состояний активных рекомендаций экспертов и их завершение (включая начисление или списание рейтинга эксперта) в случае достижения одного из параметра завершения рекомендации: тейкпрофита, стоплосса или завершения периода.
- «Сервис эмуляции биржи» – сервис, эмулирующий биржу, специализирующуюся на операциях с наиболее известной криптовалютой «Bitcoin». Предоставляет API для получения текущего курса криптовалюты.
- «Сервис эмуляции времени» – сервис предоставляет API для получения «текущей» даты в периоде с 18.08.2010 по 26.04.2018, а также API для смены «текущей» даты.

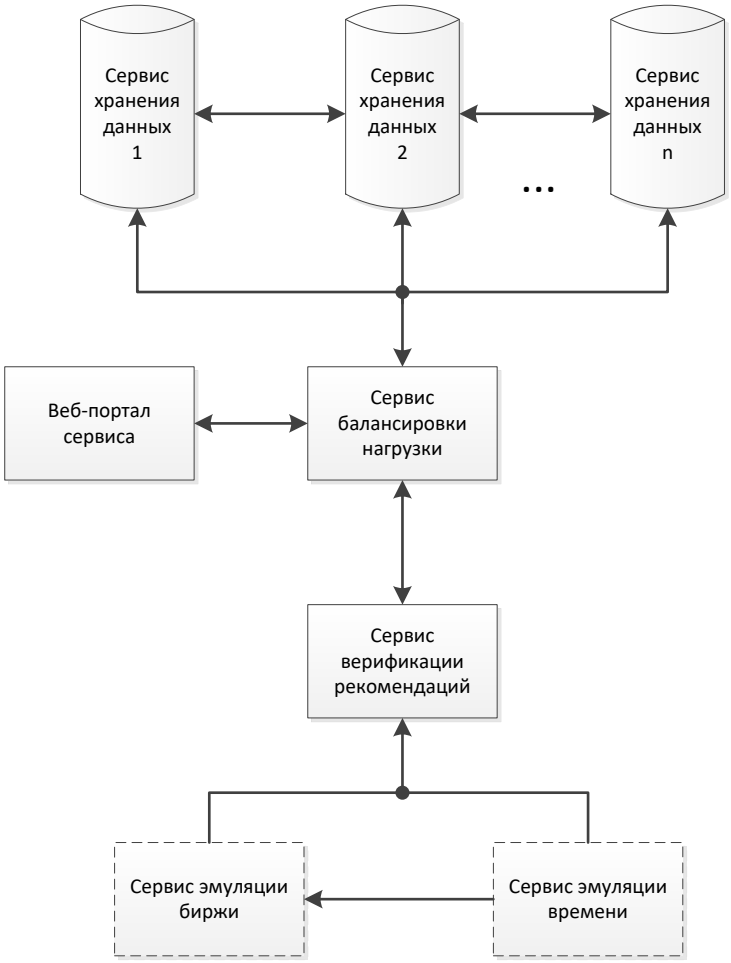

Рисунок 8 – архитектура системы

«Сервис балансировки нагрузки» – сервис, реализующий логику распределения запросов от клиентов системы к сервисам хранения данных. В качестве основополагающего фактора балансировки нагрузки путем распределения запросов в рамках научно-исследовательской работы выбран циклический итерационный метод, поочередно перенаправляющий запросы к каждому из **доступных на данный момент** сервисов хранения данных, при условии **согласованности данных**. В противном случае – минуя вопрос производительности системы, запросы будут направлены к сервису хранения данных с **актуальной информацией**. Сервис позволяет реализовывать логику балансировки нагрузки любой сложности с привлечением сторонних средств мониторинга и анализа производительности текущих параметров

информационной системы в целом (пропускной способности каналов между теми или иными сервисами, скоростью чтения данных с диска и т. п.).

«Сервис хранения данных» – сервис, непосредственно взаимодействующий с хранилищем данных, возвращающий их по запросам. Схема структуры хранилища данных представлена следующим образом (рисунок 9):

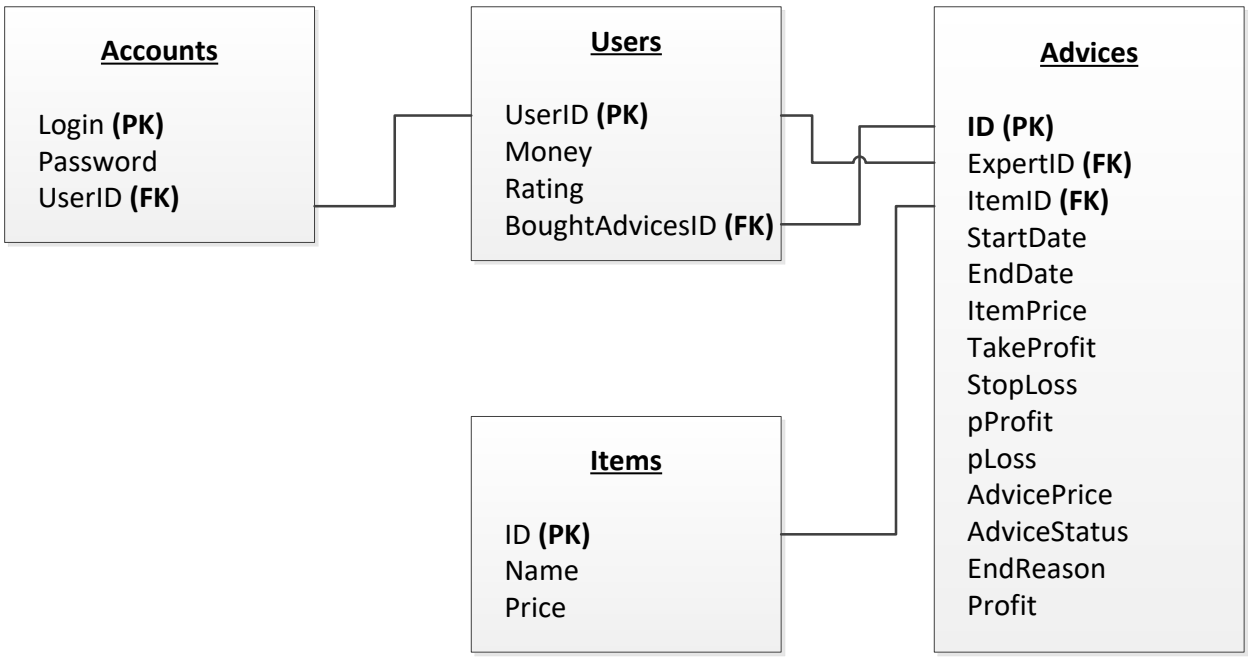

Рисунок 9 – структура данных хранилища системы

- Таблица «**Accounts**» – описывает сущность «учетная запись»;
  - Login – логин пользователя системы;
  - Password – md5 хэш пароля пользователя;
  - UserID – уникальный идентификатор пользователя, является внешним ключом и ссылается на запись в таблице «Users»;

- Таблица «**Users**» – описывает сущность «пользователь сервиса»;
  - UserID – уникальный идентификатор пользователя, является первичным ключом;

- Money – состояние счета пользователя;
- Rating – рейтинг эксперта;
- BoughtAdvicesID – приобретенные пользователем советы – список идентификаторов, является внешним ключом и ссылается на записи таблицы «Advices»;

- Таблица «**Items**» – описывает сущность «финансовый инструмент»;
  - UserID – уникальный идентификатор финансового инструмента, является первичным ключом;
  - Name – название финансового инструмента;
  - Price – текущая стоимость финансового инструмента;

- Таблица «**Advices**» – описывает сущность «рекомендация эксперта»;
  - ID – уникальный идентификатор рекомендации эксперта, является первичным ключом;
  - ExpertID – уникальный идентификатор пользователя – эксперта, зарегистрировавшего рекомендацию, является внешним ключом и ссылается на запись в таблице «Users» ;
  - ItemID – уникальный идентификатор финансового инструмента, является внешним ключом и ссылается на запись в таблице «Items»;
  - StartDate – дата начала рекомендации;
  - EndDate – дата завершения рекомендации;
  - ItemPrice – цена покупки финансового инструмента;
  - TakeProfit – цена продажи финансового инструмента. При этом будет зафиксирован доход;

- StopLoss – цена продажи финансового инструмента. При этом будет зафиксирован убыток;
- pProfit – максимальная возможная прибыль;
- pLoss – максимальный возможный убыток;
- AdvicePrice – стоимость рекомендации;
- AdviceStatus – текущий статус рекомендации.
- EndReason– причина завершения рекомендации.
- Profit – итоговая прибыль \ убыток.

Может располагаться, как на отдельно взятой машине с хранилищем данных, так и совместно с другими частями информационной системы. Между всеми сервисами хранения данных системы реализуются механизмы репликации, позволяющие максимально эффективно поддерживать согласованность данных.

Репликация представляет собой набор технологий копирования и распространения данных и объектов между хранилищами данных, а также синхронизации хранилищ данных для поддержания согласованности. Тип репликации, которая выбирается для приложения, зависит от многих факторов, в том числе от физической среды репликации, типа и объема реплицируемых данных, а также от того, обновляются данные на подписчике или нет. Физическая среда включает в себя количество и расположение компьютеров, участвующих в репликации, и зависит от того, являются эти компьютеры клиентами (рабочие станции, переносные или карманные компьютеры) или серверами.

Репликация любого типа обычно начинается с начальной синхронизации опубликованных объектов между издателем и подписчиками. Эта начальная синхронизация может быть выполнена репликацией при помощи моментального снимка, который является копией всех объектов и

данных, заданных публикацией. После создания моментального снимка он доставляется подписчикам.

Для некоторых приложений репликация моментальных снимков полностью покрывает их потребности. Для приложений других типов важно, чтобы последующие изменения данных доставлялись подписчику дополнительными порциями с течением времени. Для некоторых приложений также нужно, чтобы изменения переносились также от подписчика обратно к издателю. Такие приложения могут использовать как репликацию транзакций, так и репликацию слиянием.

Репликация транзакций обычно используется в сценариях «сервер-сервер», для которых необходима высокая пропускная способность, в том числе улучшение масштабируемости и доступности, хранение и протоколирование данных, интеграция данных с нескольких сайтов, объединение разнородных данных, автономная обработка пакетов.

В рамках разрабатываемой системы наиболее эффективный вариант представляет репликация транзакций, являющаяся основой сервиса хранения данных и сервиса балансировки нагрузки.

Архитектура системы предусматривает возможность быстрого масштабирования подсистемы хранения данных, в зависимости от возросших нагрузок или проблем отказоустойчивости.

Сервисы, выделенные на схеме пунктирным контуром («Сервис эмуляции биржи» и «Сервис эмуляции биржи»), носят временный характер и реализованы исключительно в целях демонстрации работы интерактивной системы коллективного управления инвестиционным пакетом.

«Сервис эмуляции биржи» – внешний (по отношению к интерактивной системе коллективного управления инвестиционным пакетом) сервис, эмулирующий биржу, специализирующуюся на операциях с наиболее

известной криптовалютой «Bitcoin». Предоставляет API для получения «текущего» курса криптовалюты. Имеет собственное хранилище данных, содержащее все посуточные (на 00:00) значения курса криптовалюты «Bitcoin» в периоде с 18.08.2010 по 26.04.2018. Данный выбор периода обусловлен тем фактом, что курс криптовалюты «Bitcoin» 18.08.2010 впервые превысил отметку стоимости в 0$ и составил 0,74$.

«Сервис эмуляции времени» – внешний (по отношению к интерактивной системе коллективного управления инвестиционным пакетом) сервис, эмулирующий текущую дату в периоде с 18.08.2010 по 26.04.2018.

4 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

4.1 ОБОСНОВАНИЕ ВЫБОРА СРЕДСТВ И ТЕХНОЛОГИИ ДЛЯ РЕАЛИЗАЦИИ ПОСТАВЛЕННОЙ ЗАДАЧИ

Данный этап выполнения научно-исследовательской работы, направленной на разработку интерактивной системы коллективного управления инвестиционным пакетом, заключался в выборе средств и технологий для реализации поставленной задачи.

Так как разработка системы носит не академический характер и направлена на развёртывание и практическое использование в сегменте услуг бизнес-аналитики фондового рынка, одним из основных критериев при выборе средств и технологий стал вопрос об использовании бесплатного или свободного программного обеспечения.

В качестве основой технологии для реализации веб-сервисов была выбрана технология «node.js», обладающая лицензией X11 – одной из лицензий открытого программного обеспечения, разработанной Массачусетским технологическим институтом.

Node или Node.js — современная программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно в backend-е, выполняя роль веб-сервера. Однако в рамках научно-исследовательской работы клиентская сторона пользовательского интерфейса также реализуется технологией Node.js.

В качестве основы для реализации хранилища данных интерактивной системы коллективного управления инвестиционным пакетом был выбран формат TSV – (англ. *tab separated values* — значения, разделённые табуляцией) – текстовый формат для представления таблиц баз данных. Формат позволяет хранить информацию, организованную как списки с разделенными полями. Каждая запись в таблице — это строка текстового файла. Каждое поле записи отделяется от других с помощью символа табуляции.

Это решение позволило на данном этапе разработки сервиса абстрагироваться от выбора определенной системы управления базой данных, так как процесс представляет собой сложную многопараметрическую задачу и является одним из важных этапов при разработке приложения. Наиболее рациональный подход к данному вопросу представлен следующей стратегией: определить тип СУБД (реляционная или не реляционная) и саму СУБД после завершения разработки системы и проведения нагрузочного тестирования, которое позволит определить наиболее уязвимые к нагрузкам места системы.

4.2 ВЫБОР МЕТОДОЛОГИИ РАЗРАБОТКИ

Выбор методологии разработки зависит от специфики проекта, бюджета, а также субъективных предпочтений.

Процесс разработки интерактивной системы коллективного управления инвестиционным пакетом был выстроен в соответствии с «Инкрементной» методологией разработки программного обеспечения (рисунок 10). Данная методология подразумевает деление полных требований к системе на различные сборки. Терминология часто используется для описания поэтапной сборки программного обеспечения. Процессу разработки

свойственно иметь несколько циклов разработки – вместе они составляют жизненный цикл «мульти-водопад».

Цикл разработки системы разделен на более мелкие быстро реализуемые модули. Каждый модуль системы поочередно проходит фазы формирования требований, проектирования, кодирования, внедрения и тестирования. Процесс разработки системы на основе инкрементной модели предполагает создание на первом этапе продукта в базовой конфигурации, обладающего минимально-необходимым функционалом, а далее – уже последовательное добавление нового функционала, так называемых «инкрементов». Процесс разработки продолжается до формирования полной системы со всем требуемым функционалом и бизнес-логикой.

Рисунок 10 – инкрементная методология

Использование методологии инкрементной разработки оправдано в случае, если:

- основные требования к системе четко определены и понятны. В то же время некоторые детали могут дорабатываться с течением времени;
- требуется максимально ранний вывод продукта на рынок;
- планируется внедрение набора рисковых фич или целей.

Данные факты неоспоримо подтверждают правильность выбора методологии инкрементной разработки для создания интерактивной системы коллективного управления инвестиционным пакетом.

Во-первых, основные требования к системе были сформированы на этапе описания бизнес-процессов системы, однако этапы построения архитектуры и выбора средств и технологий реализации системы привнесли некоторые корректировки в данные требования.

Во-вторых, справедливо, что для разрабатываемой системы требуется максимально ранний вывод продукта на рынок – такие факты, как отсутствие конкурентов и новизна идеи не могут долго оставаться объективной реальностью, так как ниша фондового рынка популярна как никогда прежде, и активно штурмуется разработчиками программного обеспечения.

В-третьих, описываемый вариант системы представляет собой базовый вариант функционала сервиса, однако предполагается дальнейшая доработка системы и внедрение принципиально новых механизмов бизнес-логики.

4.3 ОПИСАНИЕ МОДУЛЕЙ И ФУНКЦИЙ

Модуль: Веб-портал сервиса

Файл: *server.js*

Функции:

1. *Start* – функция выполняет начальную загрузку и запуск веб-сервиса.

Файл: *router.js*

Функции:

1. *Route* – функция задает соответствия между различными HTTP-запросами и соответствующими частями кода обработчиков.

Файл: *verApi.js*

Функции:

1. *userVer* – функция выполняет перенаправление запроса на проверку подлинности и соответствие введенных учетных данных пользователя для доступа к веб-порталу сервиса к сервису балансировки нагрузки, а затем – к соответствующему сервису хранения данных, и, в соответствии с полученным результатом проверки выполняет возврат клиенту html страницы.
2. *setStatus* – функция выполняет изменение в хранилище данных информации о текущем статусе активности пользователя в рамках веб-портала сервиса.
3. *checkStatus* – функция выполняет проверку текущего статуса активности пользователя в рамках веб-портала сервиса.

Файл: *requestHandlers.js*

Функции:

1. *start* – функция выполняет возврат клиенту html страницы, содержащей форму для аутентификации пользователя при первичном обращении к веб-порталу.
2. *registration* – функция выполняет возврат клиенту html страницы, содержащей форму для регистрации в сервисе новой учетной записи пользователя.
3. *getLogo*– функция выполняет возврат статических данных – хранящейся на сервере иконки сервиса.
4. *getUserIcon* – функция выполняет возврат статических данных – хранящегося на сервере изображения пользователя.
5. *getBcg* – функция выполняет возврат статических данных – хранящегося на сервере background-а страниц веб-портала.
6. *advices* – функция выполняет возврат клиенту html страницы – главной рабочей области веб-портала, содержащей информацию о зарегистрированных и приобретенных советах экспертов.
7. *buyAdvice* – функция выполняет перенаправление запроса на приобретение текущим пользователем указанной рекомендации эксперта к сервису балансировки нагрузки, а затем – к соответствующему сервису хранения данных – где, при наличии на счете пользователя суммы, соответствующей стоимости рекомендации эксперта, будет выполнено добавление рекомендации в список приобретенных рекомендаций текущего пользователя.
8. *getAdvice* – функция выполняет возврат клиенту html страницы, содержащей подробную информацию о той или иной приобретенной рекомендации эксперта.

9. *newAdvice* – функция выполняет возврат клиенту html страницы, содержащей форму для регистрации новой рекомендации эксперта.
10. *saveAdvice* – функция выполняет перенаправление запроса на сохранение новой рекомендации эксперта к сервису балансировки нагрузки, а затем – к соответствующему сервису хранения данных – где будет выполнено добавление рекомендации в список зарегистрированных рекомендаций экспертов.

Модуль: Сервис балансировки нагрузки

Файл: *server.js*

Функции:

1. *Start* – функция выполняет начальную загрузку и запуск веб-сервиса.
2. *getEffictiveService* – функция реализует логику процесса балансировки нагрузки и возвращает соответствующий адрес сервиса хранения данных для маршрутизации запроса.

Файл: *router.js*

Функции:

1. *Route* – функция задает соответствия между различными HTTP-запросами и соответствующими частями кода обработчиков.

Модуль: Сервис хранения данных

Файл: *server.js*

Функции:

1. *Start* – функция выполняет начальную загрузку и запуск веб-сервиса.
2. app.get('/verifyUser') – функция выполняет проверку наличия (в хранилище, представленном TSV) и соответствия полученных в запросе учетных данных пользователя.
3. app.get('/getUserInfo') – функция выполняет загрузку из хранилища и отправку данных пользователя по запросу, содержащему уникальный идентификатор пользователя.
4. app.get('/getAdvices') – функция выполняет загрузку из хранилища и отправку предварительных данных обо всех зарегистрированных рекомендациях экспертов.
5. app.get('/getBoughtAdvices) – функция выполняет загрузку из хранилища и отправку предварительных данных обо всех приобретенных рекомендациях экспертов для указанного в запросе пользователя.
6. app.get('/saveadvice) – функция выполняет добавление в хранилище новой записи рекомендации эксперта, сформированной на основе переданных в запросе данных.
7. app.get('/buyadvice) – функция выполняет обновление указанной в запросе записи пользователя и выполняет добавление в список приобретенных рекомендаций – указанную в запросе. Также реализует механизм управления расчетными операциями – списание средств со счета клиента-инвестора и начисление их эксперту – автору рекомендации.
8. app.get('/getadvice) – функция выполняет загрузку из хранилища и отправку полных данных об указанной в запросе приобретенной (текущим пользователем ранее) рекомендации эксперта.

Модуль: Сервис верификации рекомендаций

Файл: *server.js*

Функции:

1. *Start* – функция выполняет начальную загрузку и запуск веб-сервиса.
2. *CheckData* – функция выполняет мониторинг состояний активных рекомендаций экспертов и их завершение (включая начисление или списание рейтинга эксперта) в случае достижения одного из параметра завершения рекомендации: тейкпрофита, стоплосса или завершения периода.

Модуль: Сервис эмуляции биржи

Файл: *server.js*

Функции:

1. *Start* – функция выполняет начальную загрузку и запуск веб-сервиса.
2. *GetCurrentPrice* – функция выполняет запрос к сервису эмуляции времени, и, в соответствии с ответом, загружает из хранилища курс «Bitcoin-а» на указанную дату.

Модуль: Сервис эмуляции времени

Файл: *server.js*

Функции:

1. *Start* – функция выполняет начальную загрузку и запуск веб-сервиса.
2. *app.get('/getcurrentdate')* – функция возвращает текущую эмулируемую дату.

3. *GetNextDate* – функция изменяет текущую эмулируемую дату на сутки вперед.
4. *app.get('/getnextdate)* – по запросу функция вызывает метод *GetNextDate*.

5 ТЕСТИРОВАНИЕ

На сегодняшний день существует множество методик тестирования и проверки программного обеспечения, служащих для выявления таких характеристик ПО, как: надёжность, эффективность и функциональность.

Методики тестирования принято классифицировать по следующим признакам:

– По объекту тестирования (функциональное тестирование, тестирование производительности, тестирование безопасности, и т. д.)
– По степени автоматизации (ручное тестирование, автоматизированное тестирование, полуавтоматизированное)
– По времени проведения тестирования (альфа-тестирование, регрессионное тестирование, бета-тестирование)
– По степени изолированности компонентов (модульное, интеграционное тестирование, системное)
– По степени подготовленности к тестированию (тестирование по документации, интуитивное тестирование)
– и т. д.

Тестирование интерактивной системы коллективного управления инвестиционным пакетом осуществлялось с целью выявления критических ситуаций, приводящих к некорректной или замедленной работе.

Одной из важнейших частей процесса разработки приложения стало *модульное* тестирование, позволившее отладить отдельные слои и методы приложения, такие как:

- работа веб-портала;

- функционирование системы управления хранилищем данных;
- реализация логики маршрутизации и распределения нагрузки;
- эмуляция биржи криптовалюты;
- эмуляция времени системы;

На следующем этапе осуществлялось интеграционное тестирование, в процессе которого были протестированы интерфейсы и связи между слоями и компонентами сервиса.

Системное тестирование стало завершающим этапом тестирования интерактивной системы коллективного управления инвестиционным пакетом. Данный этап позволил проверить соответствие законченного продукта сформированным ранее требованиям.

Подготовка демонстрационного стенда включала в себя формирование исходных данных. Был выполнен анализ поведения курса криптовалюты «Bitcoin» в периоде с 18.08.2010 по 26.04.2018 (рисунок 11) – данный выбор периода обусловлен тем фактом, что курс криптовалюты «Bitcoin» 18.08.2010 впервые превысил отметку стоимости в 0$ и составил 0,74$.

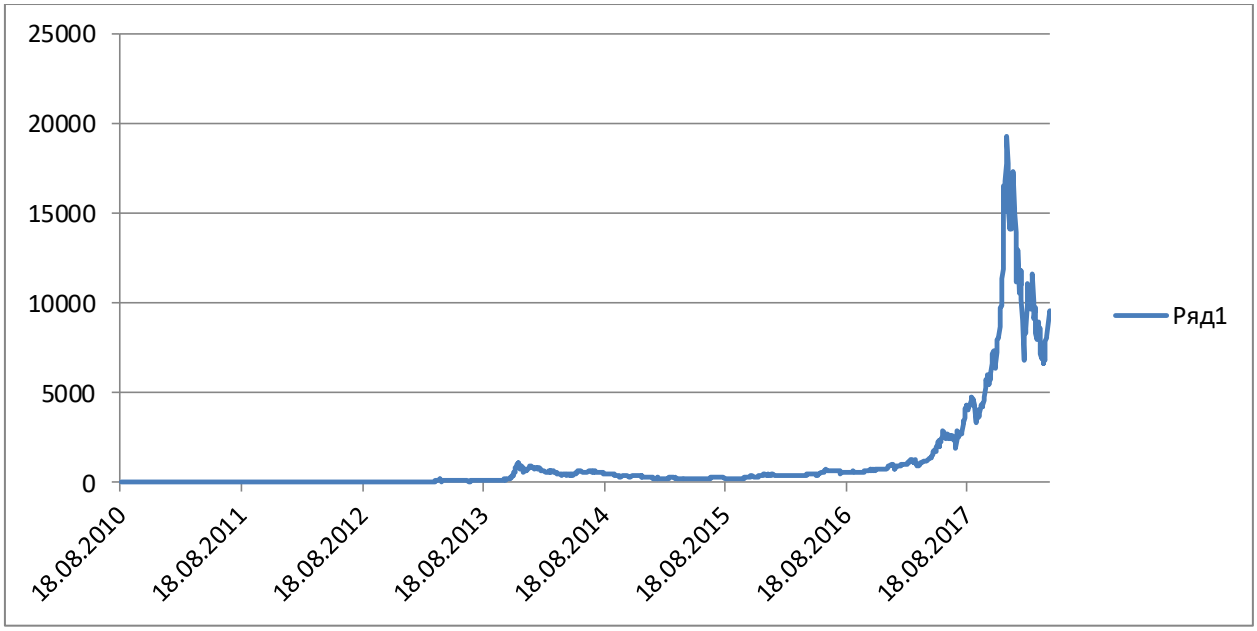

Рисунок 11 – график курса криптовалюты «Bitcoin»

В результате анализа, принимая во внимание необходимость максимальной наглядности, в качестве основы для исходных данных выбран период с 16.01.2018 по 01.02.2018 (рисунок 12).

Выбор обусловлен тем фактом, что на протяжении всего периода с высокой частотой наблюдается чередование этапов подъема и падения стоимости финансового инструмента. При этом – перепады обладают высокой амплитудой (вплоть до 2000$ в течение суток).

Экстремумы курса криптовалюты «Bitcoin» в данном периоде представлены следующим образом:

- начальная стоимость составила 11180$;
- максимальная – 12950$ (20.01.2018);
- минимальная – 9083$ (01.02.2018).

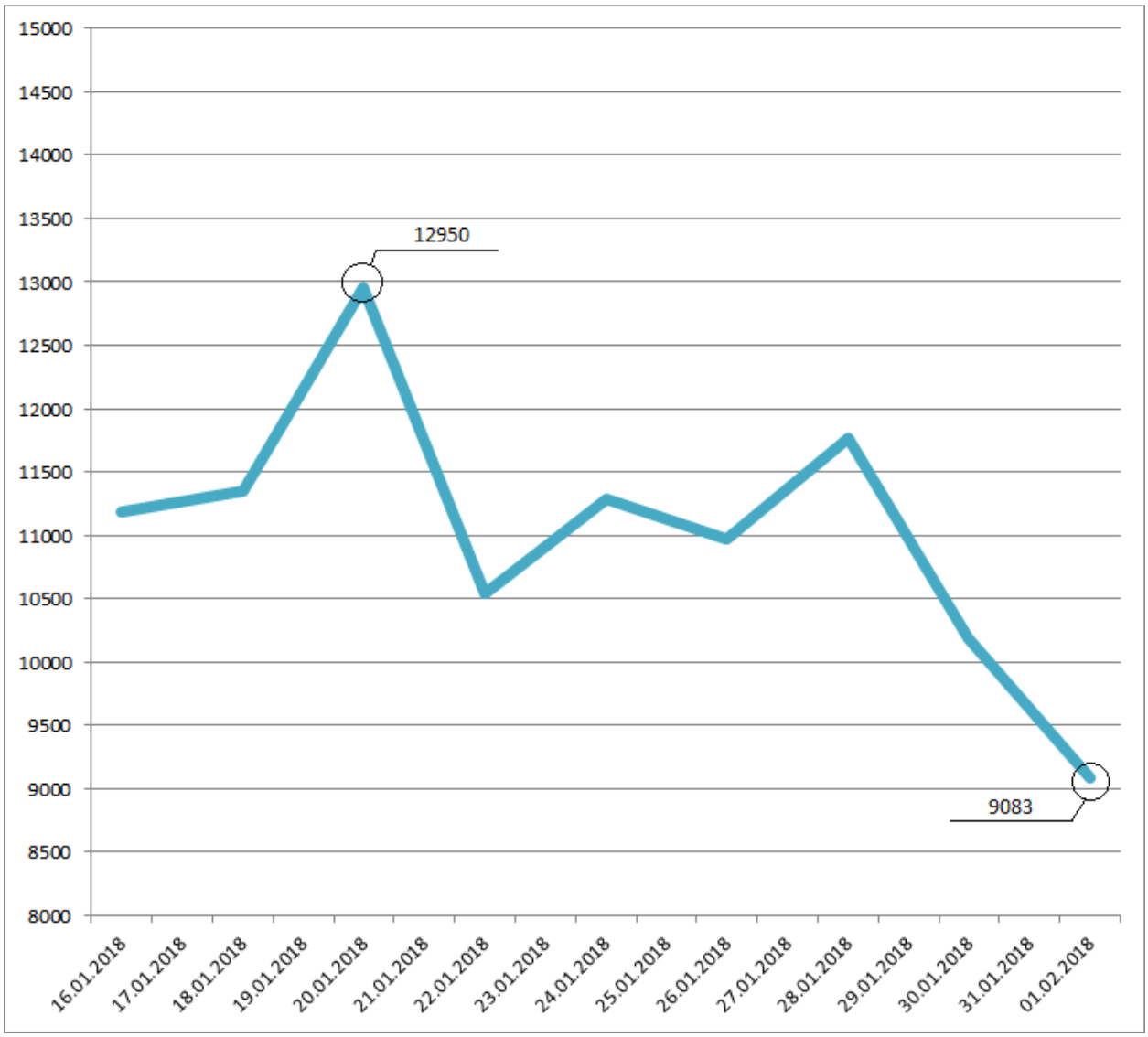

Рисунок 12 – график курса криптовалюты «Bitcoin» (период для демонстрации)

Для демонстрации корректности и полноценности реализации системы осуществлялась имитация реальной работы и функционирования бизнес-процессов. В качестве примера рассмотрим использование интерактивной системы коллективного управления инвестиционным пакетом в периоде с 16.01.2018 по 01.02.2018.

В системе эксперты представлены следующими пользователями:

- Expert-1 (рейтинг эксперта «15», состояние счета – 100$);
- Expert-2 (рейтинг эксперта «15» , состояние счета – 100$);
- Expert-3 (рейтинг эксперта «15» , состояние счета – 100$);

В соответствии с бизнес-процессом, описанным в параграфе 1.2 «Бизнес-процесс функционирования системы с точки зрения клиента-эксперта», на первом этапе эксперты формируют прогнозы на основе своего опыта или наличия инсайдерской информации (рисунок 13).

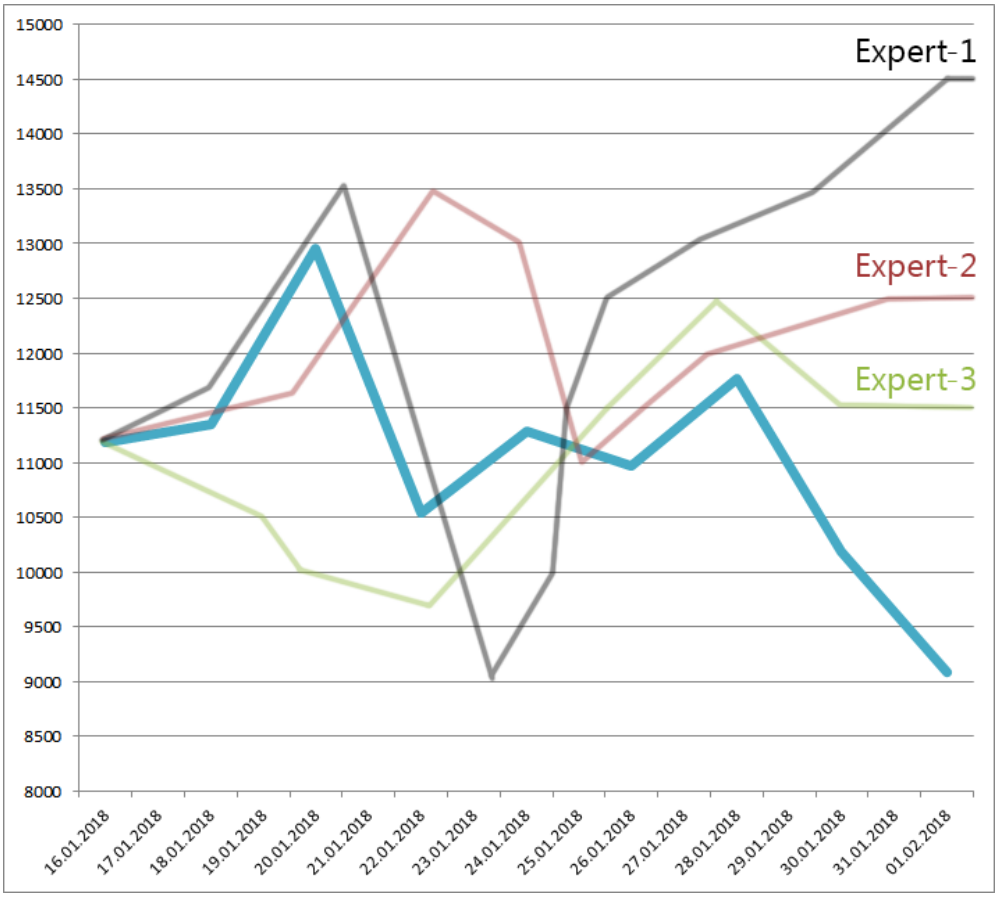

Рисунок 13 – график прогнозов курса криптовалюты «Bitcoin»

Так Expert-1 предполагает уверенный рост стоимости финансового инструмента в периоде с 16.01.2018 по 20.01.2018 до уровня 13500$, дальнейшее кратковременное стремительное падение стоимости до 9000$ в периоде с 20.01.2018 по 24.01.2018 и последующий рост стоимости до уровня 14500$ вплоть до конца периода прогноза.

В свою очередь, Expert-2 предполагает постепенный рост стоимости финансового инструмента в периоде с 16.01.2018 по 22.01.2018 до уровня 13500$, дальнейшее постепенное снижение стоимости до 11000$ в периоде с

22.01.2018 по 25.01.2018 и дальнейший плавный рост стоимости до уровня 12500$ до конца периода прогноза.

Expert-3 предполагает плавное ступенчатое снижение стоимости финансового инструмента в периоде с 16.01.2018 по 22.01.2018 до уровня 9700$, а затем – уверенный рост до 12500$ в периоде с 22.01.2018 по 28.01.2018, далее – равномерное снижение стоимости до уровня 11500$, и удержание обозначенного уровня до конца периода прогноза.

На втором этапе бизнес-процесса эксперты регистрируют сформированные прогнозы в интерактивной системе коллективного управления инвестиционным пакетом (рисунок 14 – 17).

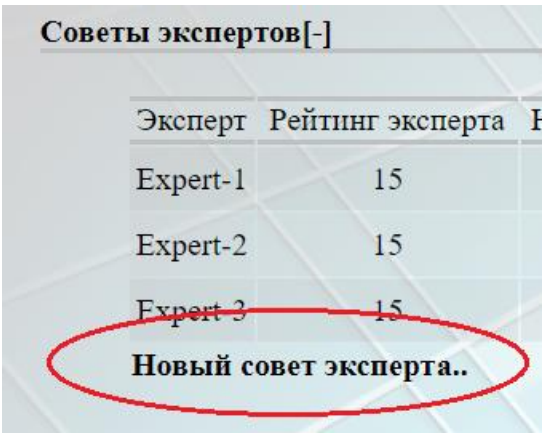

Рисунок 14 – пользовательский интерфейс (регистрация совета эксперта)

- Expert-1

  Цена покупки финансового инструмента: 11180$;

  Тейк профит: 14500$;

  Стоп лосс: 9000$;

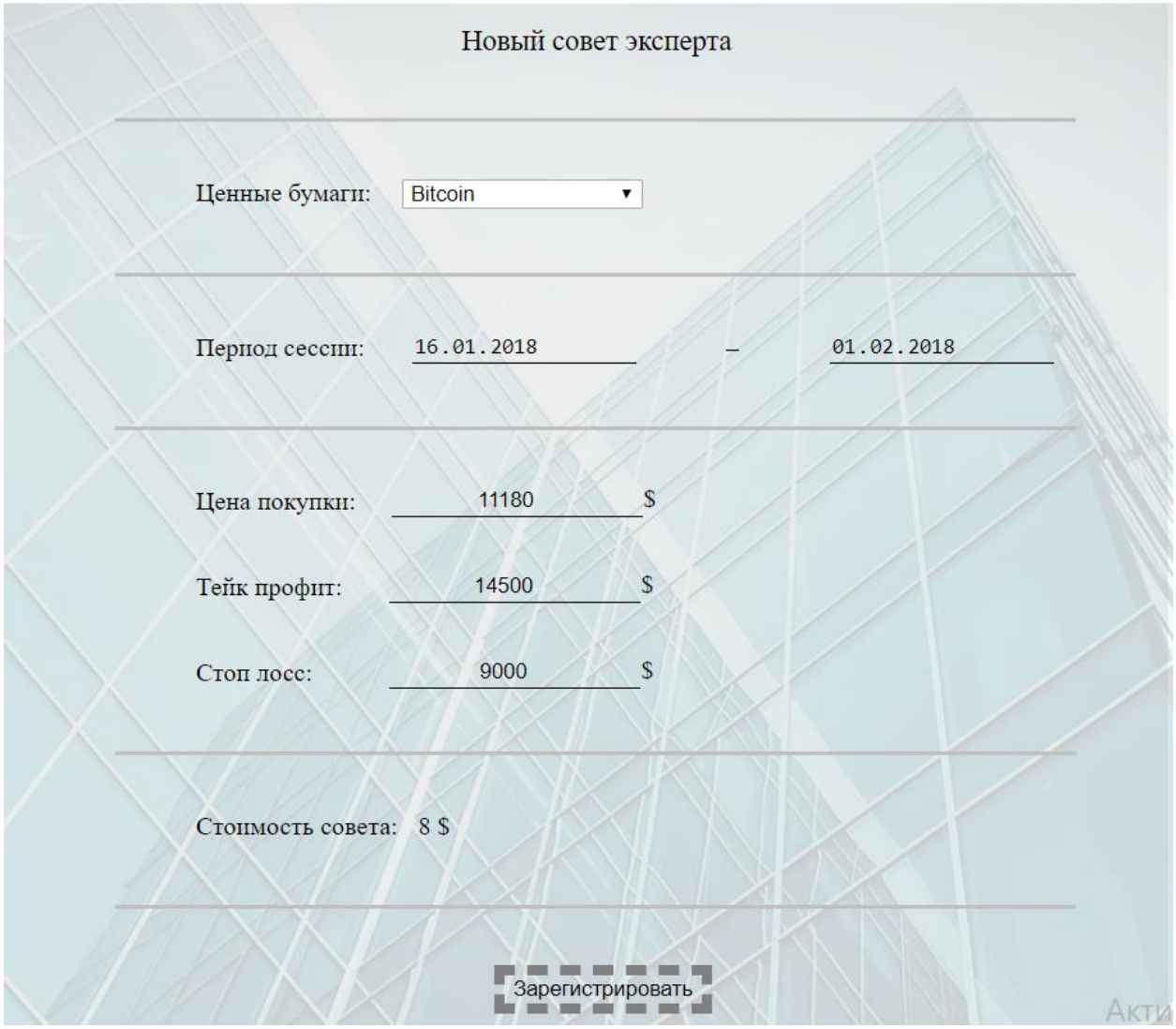

Рисунок 15 – регистрация совета эксперта (1)

- Expert-2

  Цена покупки финансового инструмента: 11180$;

  Тейк профит: 13500$;

  Стоп лосс: 11000$;

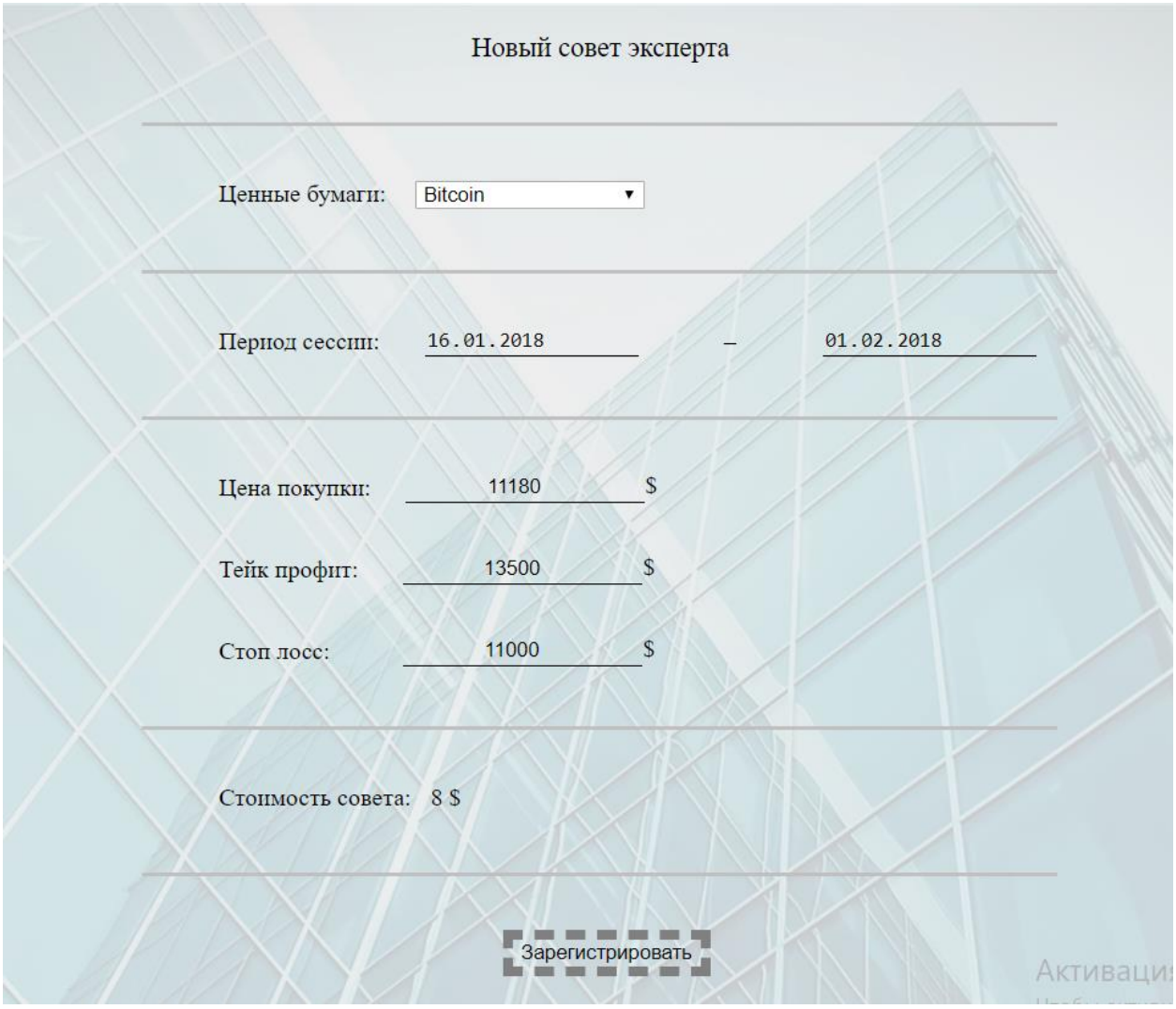

Рисунок 16 – регистрация совета эксперта (1)

- Expert-3

  Цена покупки финансового инструмента: 11180$;

  Тейк профит: 12500$;

  Стоп лосс: 10000$;

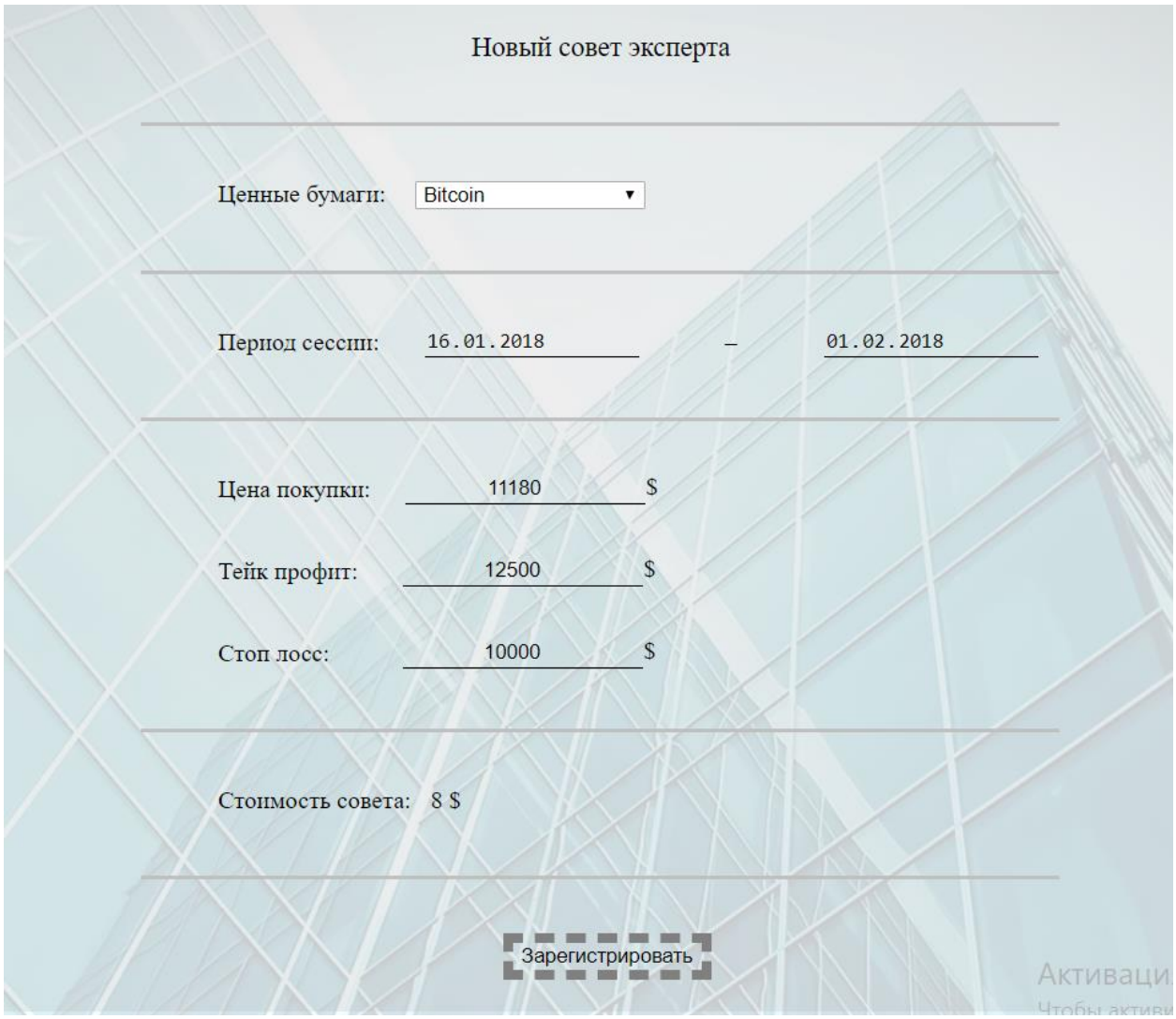

Рисунок 17 – регистрация совета эксперта (3)

Таким образом, в результате завершения второго этапа бизнес-процесса в сервисе зарегистрированы три различных рекомендации экспертов (рисунок 18) продолжительностью 13 суток и:

1. Ожидаемой прибылью – 29.7%, возможным убытком – 19,5%;
2. Ожидаемой прибылью – 20.75%, возможным убытком – 1,61%;
3. Ожидаемой прибылью – 11,81%, возможным убытком – 10,55%;

Советы экспертов[-]

Эксперт	Рейтинг эксперта	Начало сессии	Окончание сессии	Продолжительность сессии (дней)	Ожидаемая прибыль (%)	Возможная потеря (%)	Стоимость совета
Expert-1	15	2018.01.16	2018.02.01	13	29.7	19.5	8
Expert-2	15	2018.01.16	2018.02.01	13	20.75	1.61	8
Expert-3	15	2018.01.16	2018.02.01	13	11.81	10.55	8

Новый совет эксперта..

Рисунок 18 – рабочая область «Советы экспертов»

На третьем этапе бизнес-процесса в роли механизмов выступают:

- Сервис – система коллективного управления инвестиционным пакетом;
- Клиент-инвестор (user-1);

Данный этап является связующим звеном бизнес-процессов эксперта и клиента-инвестора: третий этап бизнес-процесса, реализуемого с точки зрения эксперта – «Ожидать завершения периода рекомендации», где в роли одного из механизмов выступает клиент-инвестор, пересекается с третьим этапом бизнес-процесса, реализуемого с точки зрения клиента-инвестора – «Приобрести рекомендацию эксперта», где в роли одного из механизмов выступает разработанный сервис, предоставляющий рекомендации экспертов.

Таким образом, клиент-инвестор, имеющий на балансе личного счета 100$ в интерактивной системе коллективного управления инвестиционным пакетом, приобретает рекомендации экспертов. Выполняется списание средств со счета клиента и начисление на счет инвестора (рисунок 19):

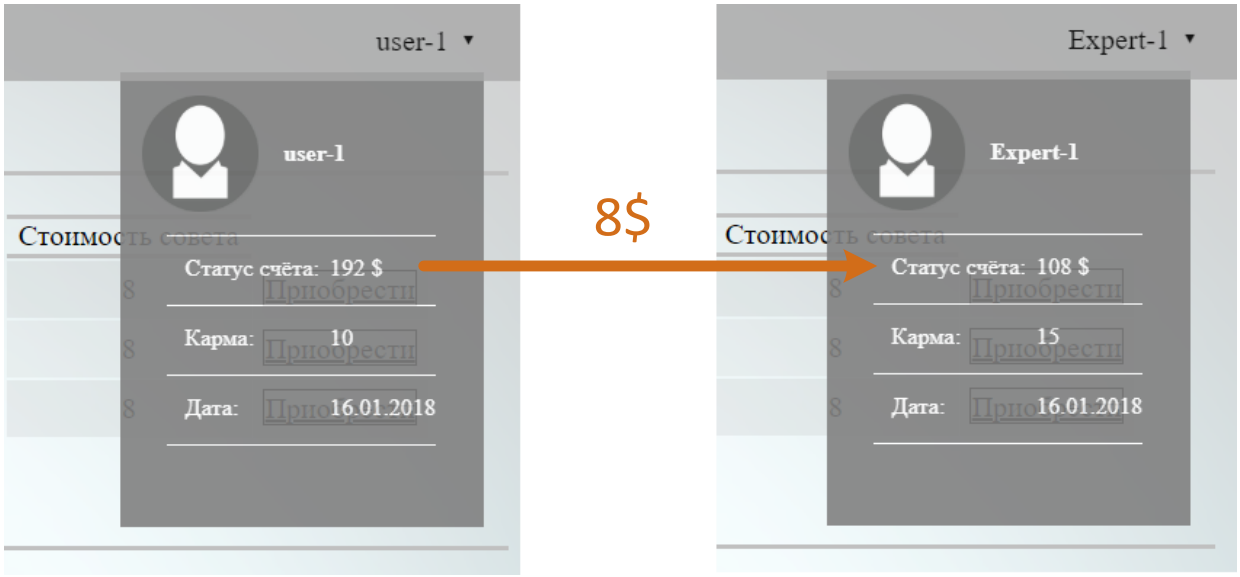

Рисунок 19 – процесс оплаты при покупке рекомендации эксперта

В результате операции клиент получает доступ к полному содержанию рекомендации эксперта (рисунок 20, 21):

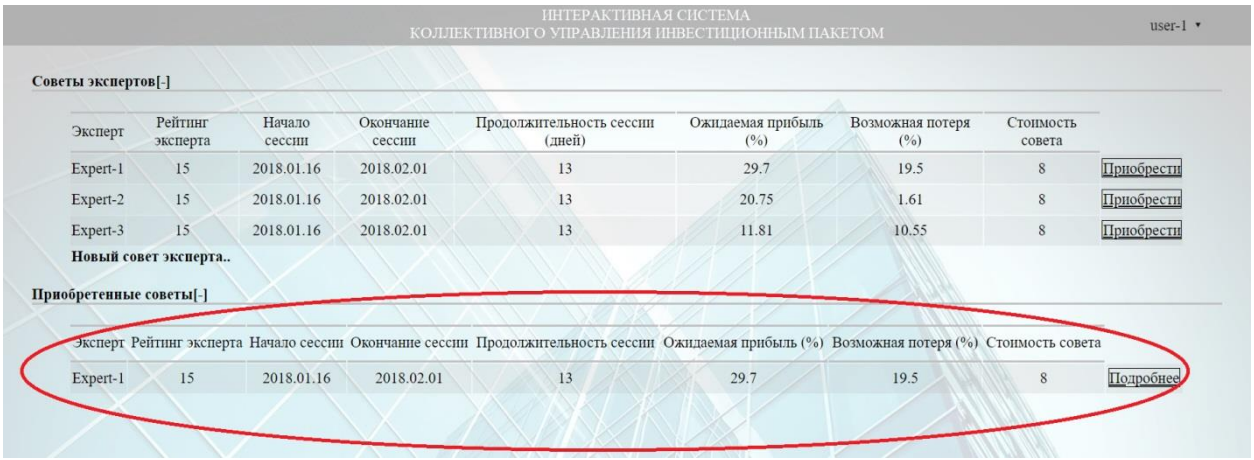

Рисунок 20 – рабочая область «Приобретенные советы»

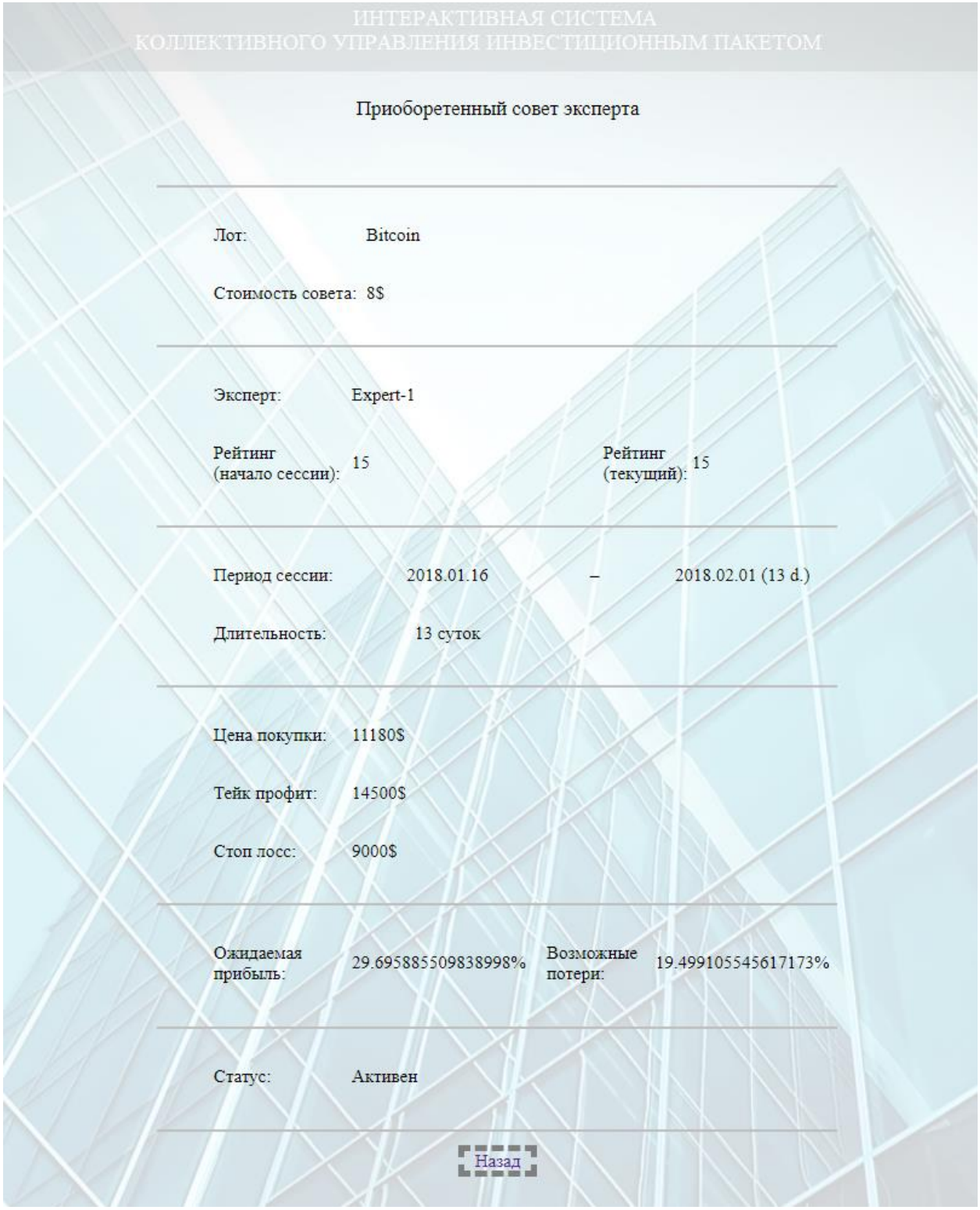

Рисунок 21 – приобретенная рекомендация

На завершающем этапе бизнес-процесса система коллективного управления инвестиционным пакетом выполняет мониторинг состояний активных рекомендаций экспертов и их завершение (включая начисление

или списание рейтинга эксперта) при достижении одного из параметров завершения рекомендации: тейкпрофита, стоплосса или завершения периода.

Таким образом, для пользователя Expert-1 справедливо следующее (рисунок 22):

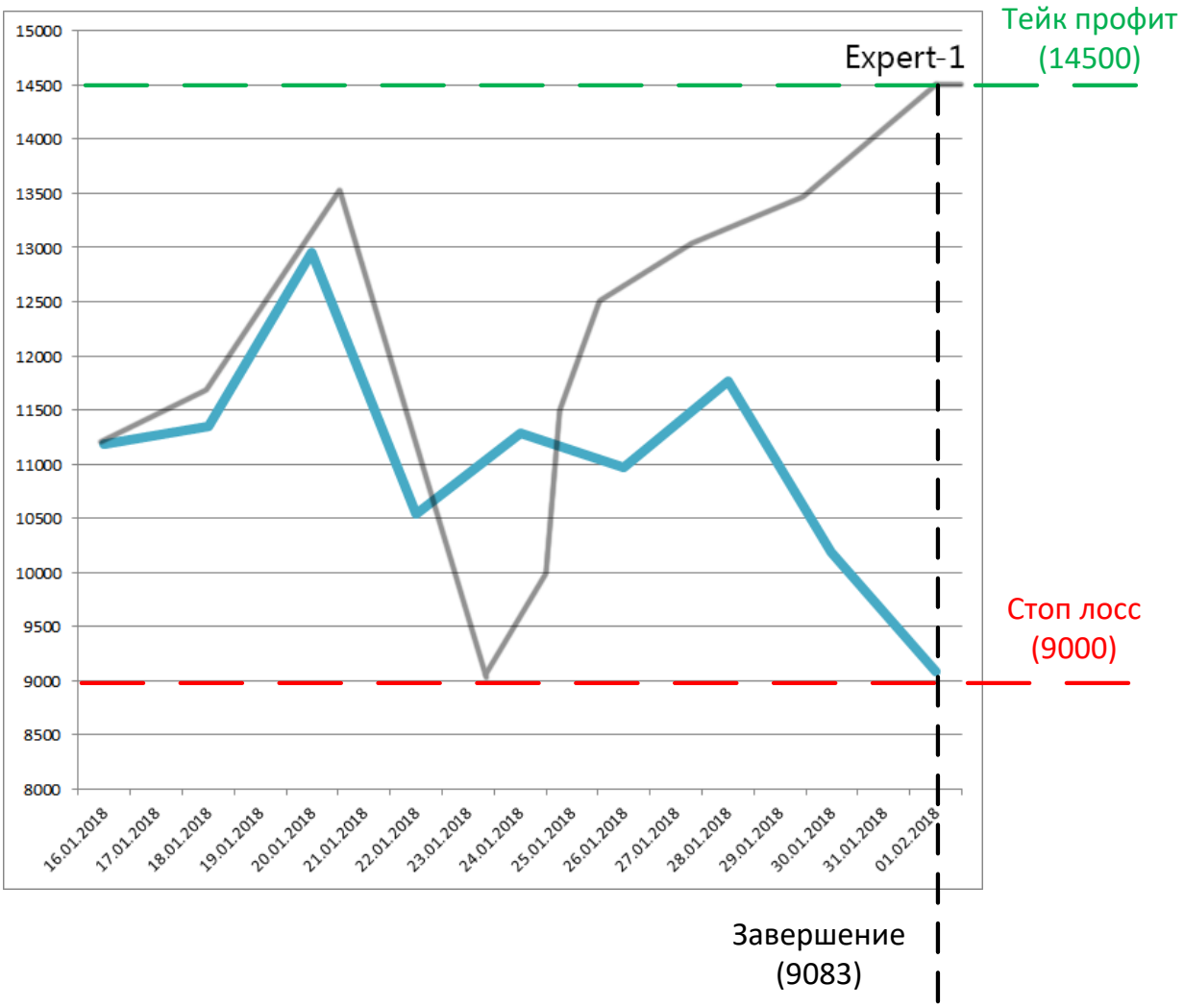

Рисунок 22 – результаты рекомендации (1)

В течение периода действия рекомендации эксперта «Expert-1» стоимость финансового инструмента не достигла ни одного из крайних значений – рекомендация была завершена по истечению срока действия 01.02.2018. Стоимость финансового инструмента на момент завершения рекомендации составила 9083$ – таким образом, рекомендация принесла

убыток в размере 18,8%. Рейтинг эксперта системы был соответственно понижен на 0,188 и составил 9,81.

Для пользователя Expert-2 (рисунок 23):

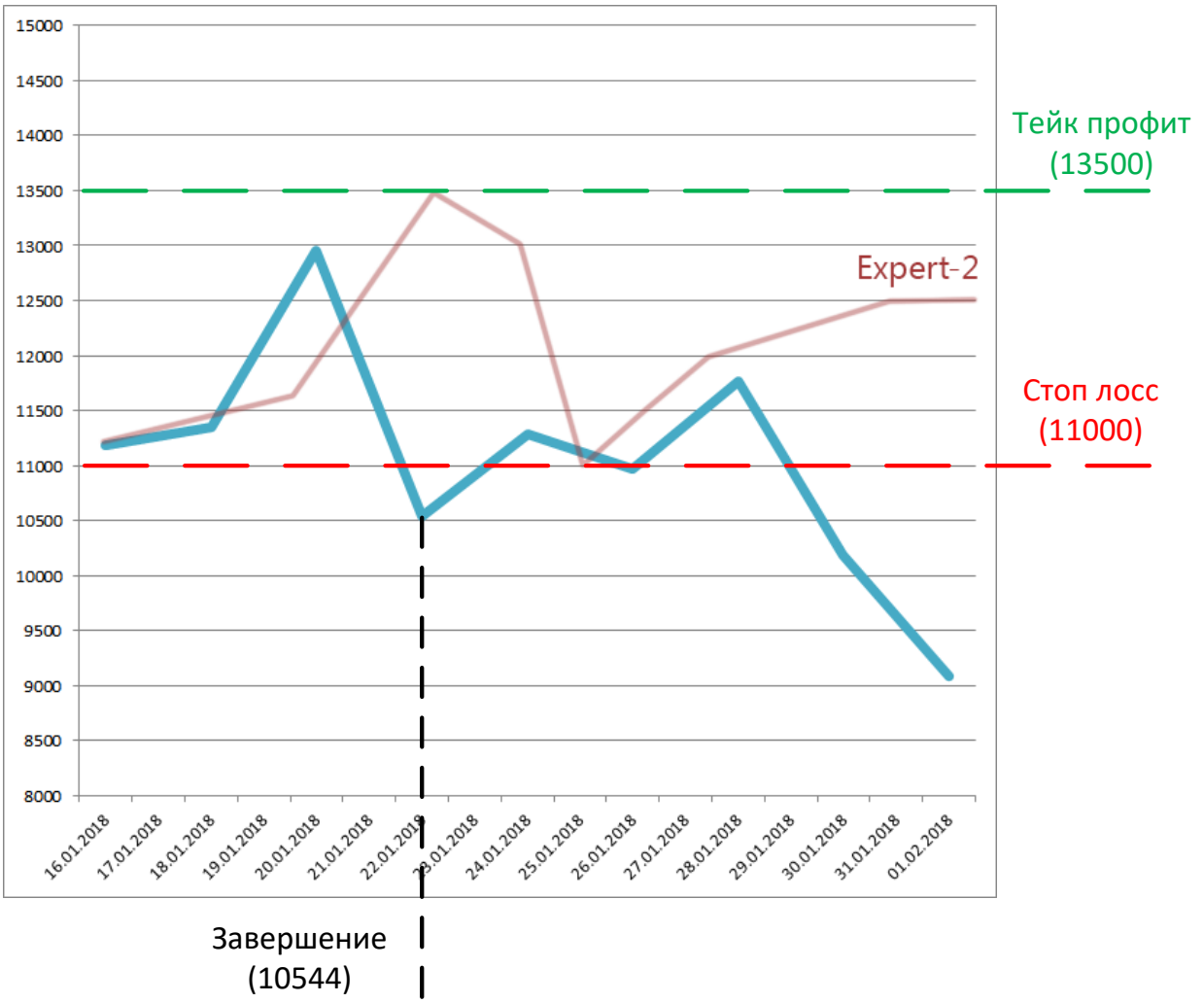

Рисунок 23 – результаты рекомендации (2)

В течение периода действия рекомендации эксперта стоимость финансового инструмента достигла отметки ниже отметки стоп лосса – 22.01.2018. Стоимость финансового инструмента на момент завершения рекомендации составила 10544$ – таким образом, рекомендация принесла убыток в размере 5,7%. Рейтинг эксперта системы был соответственно понижен на 0,057 и составил 9,94.

Для пользователя Expert-3 (рисунок 24):

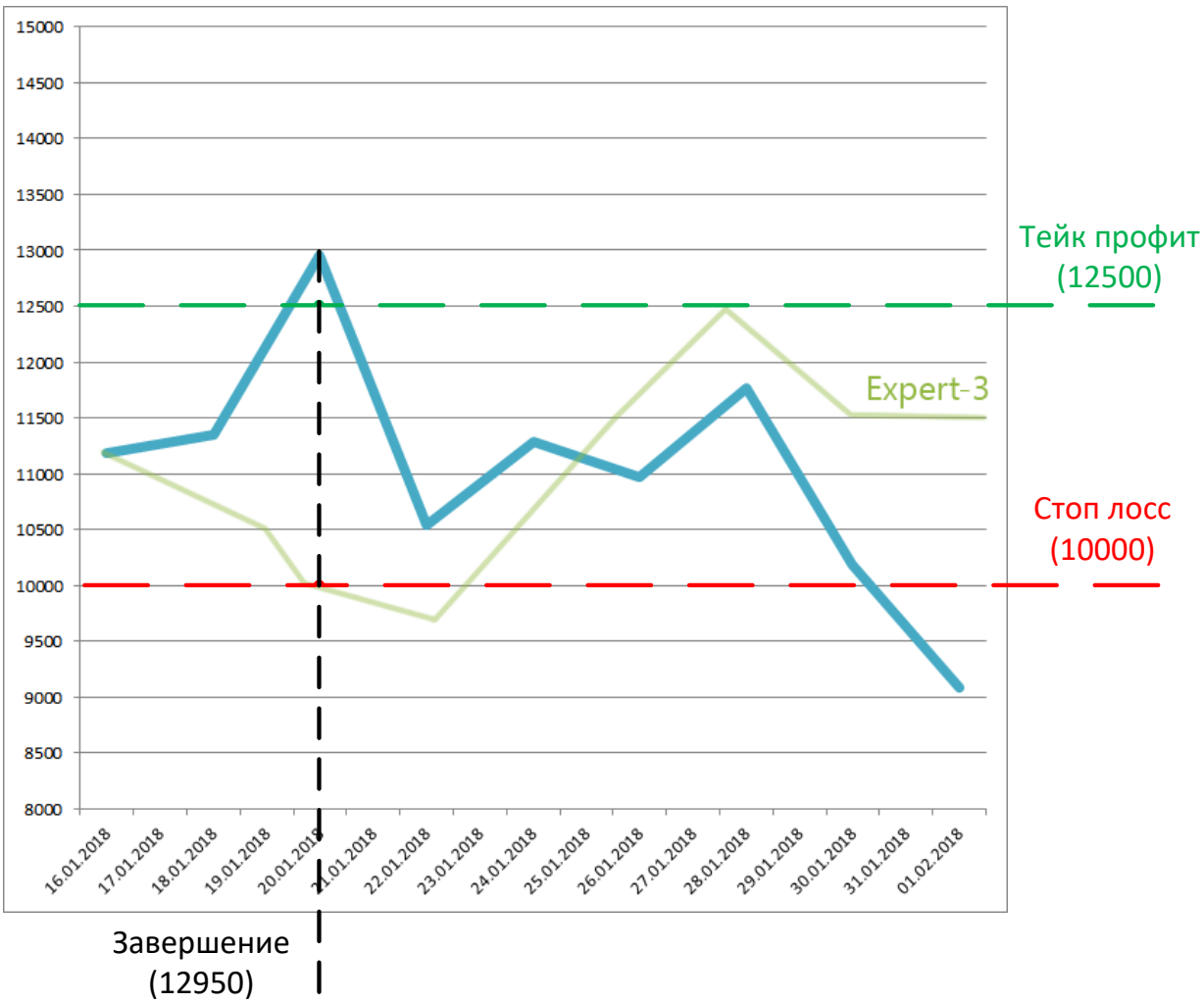

Рисунок 24 – результаты рекомендации (3)

В течение периода действия рекомендации эксперта стоимость финансового инструмента достигла отметки выше отметки тейк профита – 20.01.2018. Стоимость финансового инструмента на момент завершения рекомендации составила 12950$ – таким образом, рекомендация принесла прибыль в размере 15,8%. Рейтинг эксперта системы был соответственно повышен на 0,158 и составил 10,158.

Таким образом, в результате использования интерактивной системы коллективного управления инвестиционным пакетом в периоде с 16.01.2018 по 01.02.2018 следующими пользователями: Expert-1, Expert-2, Expert-3, user-1, – была воспроизведена имитация реальной работы и функционирования

четырех бизнес-процессов системы, результатом завершения которых стало следующее:

- удовлетворение спроса клиента-инвестора на приобретение бизнес-аналитики по заданному финансовому инструменту;
- получение прибыли от продажи услуг бизнес-аналитики экспертами системы;
- изменение рейтинга экспертов системы в соответствии с результатами предоставленных рекомендаций;

Из представленного выше следует, что системное тестирование выполнено успешно – интерактивная система коллективного управления инвестиционным пакетом реализована корректно и в полной мере соответствует требованиям, сформированным в разделе «Архитектура приложения».

6 НАПРАВЛЕНИЯ РАЗВИТИЯ

Дальнейшее развитие интерактивной системы коллективного управления инвестиционным пакетом предполагает решение следующих первоочередных задач.

6.1 РАЗРАБОТКА И ВНЕДРЕНИЕ МЕХАНИЗМА «КОЛЛЕКТИВНОГО УПРАВЛЕНИЯ»

Данный этап заключается в реализации системы алгоритмического анализа и автоматизированного принятия решений об инвестировании средств, внесенных клиентом-инвестором в рамках установленной им сессии, на основе набора рекомендаций экспертов и их рейтинга.

Заинтересованность в нововведении всех сторон-участников системы коллективного управления инвестиционным пакетом обусловлена следующими факторами:

- клиент-инвестор освобождается от какой-либо занятости в процессе инвестирования путем регистрации инвестиционных сессий (устанавливает такие параметры как период сессии, размер инвестиции, финансовый инструмент), однако рискует своими финансовыми ресурсами;
- эксперт системы получает возможность принимать участие в инвестиционных сессиях, созданных клиентами-инвесторами. В случае, если его рекомендация способствовала получению прибыли – получает часть, а так же дополнительный рейтинг эксперта системы. Заинтересован привнести свои рекомендации в наибольшее количество сессий. Не рискует личными

финансовыми ресурсами, однако вовлечен в постоянную занятость в рамках системы и рискует рейтингом эксперта;

- сервис заинтересован в успешных инвестиционных сессиях клиентов – а следственно в компетентных экспертах, получая процент от прибыли.

В основе механизма – алгоритм анализа набора рекомендаций экспертов, на основе установления весов (рекомендациям) по уровню рейтинга эксперта, и дальнейшее формирование итоговой стратегии инвестирования.

Выполнение данного этапа включает в себя:

- разработку структуры новых бизнес-процессов системы и формирование требований к компоненту;
- разработку архитектуры компонента;
- создание и отладку компонента в соответствии с выбранной методологией;
- внедрение компонента в существующую систему и последующее тестирование.

6.2 ИССЛЕДОВАНИЕ И ДОРАБОТКА СИСТЕМЫ С ТОЧКИ ЗРЕНИЯ ВОПРОСА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Так как в основе бизнес-процессов системы лежат механизмы управления финансовыми ресурсами, подразумевающие финансовую ответственность интерактивной системы коллективного управления инвестиционным пакетом перед своими клиентами – важнейшей задачей является обеспечение информационной безопасности всех процессов системы.

6.3 РЕАЛИЗАЦИЯ КОНЦЕПЦИИ «FRONT END – BACK END»

Интерактивная система коллективного управления инвестиционным пакетом, разработанная в рамках научно-исследовательской работы, представляет собой базовую версию целевого продукта. Данная версия реализует формирование веб-страниц, их заполнение соответствующим контентом и отправку клиенту на стороне сервера.

Современный веб-сервис должен обладать качественно проработанным, простым в использовании и интуитивно понятным интерфейсом. Планируется доработка компонентов страниц портала для реализации управления инвестиционными сессиями, внедрение механизмов мониторинга и статистики.

Для достижения поставленной задачи планируется реализация «front end-а» на основе современных фреймворков, таких как AngularJS, IONIC, ReactJS и т. п.

6.4 РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

Современный ритм жизни диктует человечеству свои условия – требует мобильности и минимизации временны**ы**х затрат на любые административные процессы. Как следствие – огромную популярность приобретают мобильные приложения, позволяющие пользоваться сервисами с повышенным удобством и эффективностью.

Разработка мобильного приложения для наиболее популярных платформ, таких как Android и iOS, является неотъемлемой частью жизненного цикла современных сервисов. Одним из основных критериев при выборе средств и технологий для реализации поставленной задачи стал вопрос об использовании бесплатного или свободного программного обеспечения.

В качестве основой технологии был выбран «Xamarin». Это фреймворк под лицензией MIT (лицензия открытого программного обеспечения, разработанная Массачусетским технологическим институтом) для кроссплатформенной разработки мобильных приложений (iOS, Android, Windows Phone) с использованием языка C#.

Фреймворк состоит из нескольких основных частей:

- Xamarin.IOS — библиотека классов для C#, предоставляющая разработчику доступ к iOS SDK;
- Xamarin.Android — библиотека классов для C#, предоставляющая разработчику доступ к Android SDK;
- Компиляторы для iOS и Android;
- IDE Xamarin Studio;
- Плагин для Visual Studio.

Конечное мобильное приложение будет включать в себя полный функционал веб-сервиса интерактивной системы коллективного управления инвестиционным пакетом, а также содержать механизмы уведомлений о достигнутых вехах в процессе жизненного цикла рекомендаций экспертов и инвестиционных сессий, о регистрации новых рекомендаций и т. д.

6.5 РЕГИСТРАЦИЯ ТОВАРНОГО ЗНАКА

Регистрация товарного знака, включающая разработку логотипа и названия, является завершающим этапом подготовки интерактивной системы коллективного управления инвестиционным пакетом к выходу на рынок предоставления услуг бизнес-аналитики в сегменте фондового рынка.

7 ЗАКЛЮЧЕНИЕ

В рамках научно-исследовательской работы, направленной на решение актуальной задачи – разработку электронной торговой площадку предоставления услуг бизнес-аналитики, была разработана *интерактивная система коллективного управления инвестиционным пакетом*.

В данной работе описана архитектура, обоснование выбора средств и технологии, процесс разработки и функционал конечного приложения, реализующего платформу для взаимовыгодного сотрудничества клиентов – лиц, заинтересованных в приобретении бизнес-аналитики по тому или иному финансовому инструменту на определенный период, и экспертов системы – пользователей, обладающих информацией, доступной узкому кругу лиц, особыми навыками или талантами в области анализа текущего состояния и прогнозирования тенденций изменения тех или иных финансовых инструментов фондового рынка.

Основной функционал сервиса представлен в виде моделей бизнес-процессов, созданных на основе методологии «IDEF0». Функциональные модели наглядно демонстрируют заинтересованность всех субъектов бизнес-процесса во взаимовыгодном сотрудничестве на базе разработанной системы.

Архитектура (системы) обеспечивает возможность балансировки нагрузки, а также способна к масштабированию и быстрому наращивания аппаратных мощностей.

Программная реализация *интерактивной системы коллективного управления инвестиционным пакетом* выполнена на основе технологии «node.js» под лицензией X11 – лицензии открытого программного обеспечения, разработанной Массачусетским технологическим институтом, и в соответствии с инкрементной методологией создания ПО,

подразумевающей несколько циклов разработки до формирования полной системы со всем требуемым функционалом и бизнес-логикой.

Система, разработанная в рамках научно-исследовательской работы, представляет собой базовую версию целевого продукта. Как результат – сформированы основные направления развития для вывода сервиса на рынок предоставления услуг бизнес-аналитики в сегменте фондового рынка.

8 СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. «Биржевой зодиак: Какие алгоритмы и инструменты применяются для прогнозирования движения цен акций / Блог компании ITinvest / Geektimes». URL : https://geektimes.ru/company/itinvest/blog/272970/ (дата обращения: 09.06.18).
2. Рынок ценных бумаг — Википедия. URL: https://ru.wikipedia.org/wiki/Рынок_ценных_бумаг (дата обращения: 09.06.18).
3. Модели разработки программного продукта. URL: http://elibrary.ru/item.asp?id=27724514 (дата обращения: 09.06.18).
4. Рынок ценных бумаг. URL: http://www.studentlibrary.ru/doc/ISBN9785394025341-SCN0000.html (дата обращения: 09.06.18).
5. Инновационные подходы к совершенствованию систем электронного хранения и обработки данных. URL: http://elibrary.ru/item.asp?id=27655090 (дата обращения: 09.06.18).
6. MSSQL server теория > репликации в SQL Server 2012. База знаний TSQL,C#,ADO.net,Asp.net MVC4 полезные заметки. URL: http://mvc4.pro/Home/Details_R/851/ (дата обращения: 09.06.18).
7. Выбор методологии разработки продукта. URL: http://mylektsii.ru/13-47580.html (дата обращения: 09.06.18).
8. Xamarin: разработка под android. URL: http://www.cyberforum.ru/android-dev/thread1364516.html (дата обращения: 09.06.18).

9. Разработка веб-приложений с использованием angular.js, node.js, mongodb на примере системы психологической поддержки студентов - участников программы «полет». URL: http://elibrary.ru/item.asp?id=28163565 (дата обращения: 09.06.18).
10. Описание бизнес-процессов в проектировании при помощи нотации IDEF0. URL: http://elibrary.ru/item.asp?id=25309212 (дата обращения: 09.06.18).
11. Программная платформа Node.js. URL: http://elibrary.ru/item.asp?id=27520560 (дата обращения: 09.06.18).
12. Прикладные информационные технологии в экономике учебное пособие Москва 2016. URL: http://dlib.rsl.ru/rsl01009000000/rsl01009459000/rsl01009459009/rsl01009459009.pdf (дата обращения: 09.06.18).
13. Биржа в условиях глобальной конкуренции. URL: http://elibrary.ru/item.asp?id=20151728 (дата обращения: 09.06.18).
14. Применение системного подхода к изучению развития биржевого рынка. URL: http://elibrary.ru/item.asp?id=25402410 (дата обращения: 09.06.18).

Приложение А. Текст программы

Модуль: Веб-портал сервиса

Файл: server.js

```
var http = require("http");
var url = require("url");


var fs = require('fs');

var mkdirp = require('mkdirp');

function start(route, handle) {

    function onRequest(request, response) {

    var postData = "";
    var pathname = url.parse(request.url).pathname;
    console.log("Request for " + pathname + " received.");

    request.setEncoding("utf8");

    request.addListener("data", function(postDataChunk) {
        postData += postDataChunk;
    });

    request.addListener("end", function () {
        route(handle, pathname, response, postData, request);
    });

    }

  http.createServer(onRequest).listen(8888);
  console.log("Web Portal Server has started.");
}

     exports.start = start;
```

Файл: router.js

```
function route(handle, pathname, response, postData, request) {
  console.log("About to route a request for " + pathname);
  if (typeof handle[pathname] === 'function') {

      if (pathname == "/delete" || pathname == "/load" ||
pathname == "/upload" || pathname == "/mysessions" || pathname
```

```
== "/advices" || pathname == "/newadvice"  || pathname ==
"/saveadvice"   || pathname == "/buyadvice" || pathname ==
"/getadvice")
      {
          handle[pathname](response, postData, request);
      }
      else
      {
          handle[pathname](response, postData);
      }

  } else {
    console.log("No request handler found for " + pathname);
    response.writeHead(404, {"Content-Type": "text/plain"});
    response.write("404 Not found");
    response.end();
  }
}

     exports.route = route;
```

Файл: verApi.js

```
var fs = require("fs");

var express = require('express');
var querystring = require('querystring');
var http = require('http');

var request = require("request");

var selfUrl = "http://localhost:8888";

var BalanceService = "http://localhost";
var BalanceService_Port = 7000;

var allData = '';


//Провекрка текущего статуса пользователя
function checkStatus(userName) {
    if (fs.existsSync(path)) {
        var currentStatus = fs.readFileSync(userName + '.login',
'utf8');
        return parseInt(currentStatus);
    }

    return 0;
}
```

```
//Изменение текущего статуса пользователя
function setStatus(userName, status) {
    fs.writeFile("./" + userName + ".login", status);
}

function someAsyncFunction(a, b, c, callback) {
    setTimeout(function () {
        callback(a, b, c);
    }, 3000)
}

//Вход
function userVer(userName, _pass, response) {
    console.log("userVer");

    var userID = '';
    var result = false;

    http.get(BalanceService + ':' + BalanceService_Port +
'/verifyUser?user=' + userName + '&pass=' + _pass, (resp) => {
        let data = '';

        // A chunk of data has been recieved.
        resp.on('data', (chunk) => {
            data += chunk;
        });

        // The whole response has been received. Print out the
result.
        resp.on('end', () => {

            console.log("resp.on end");
            userID = data;

            console.log("Verification user status1: " + userID);

            if (userID != 'null' && userID != 'error')
            {
                console.log("TRUE");

                response.writeHead(302, {
                    'Location': '/advices?user=' + userName +
'&userid=' + userID
                });
                response.end();

            }
            else
            {

                var body = '<html>' +
                           '<head>' +
```

```
                                '<script type="text/javascript">' +
                                'function(error){' +
                                      'if(error.responseText ==
"showAlert")' +
                                          'alert("ALAAAAAARM!! Please
enter correct user name and password.");' +
                                          '</script>' +
                                '<meta http-equiv="Content-Type"
content="text/html; ' +
                                'charset=UTF-8" />' +
                                '</head>' +
                                '<body>' + //border-bottom: 3px
double gray;
                                '<div style="width: 100%;
height:30px; top:0; margin-top: 24px; margin-left:48%; padding-
bottom: 10px;">' +
                                '<p>RED DROP BOX<img src="logo.png"
width="30px" height="30px"></p>' +
                                '</div>' +
                                '<div style=" border-bottom: 3px
double gray; width: 90%; height:1px; margin-left:5%"/>' +
                                '<div style=" width: 150px;
height:300px; position: absolute; top:0; bottom: 0; left: 0;
right: 0;  margin: auto;">' +
                                '<form action="/loginIn"
method="post">' +
                                '<p>Login:</p><input type="text"
name="login" rows="1" cols="20"/><br>' +
                                '<p>Password:</p><input
type="password" name="pass" rows="1" cols="20"/><br>' +
                                '<br><input style="margin-left:60px"
type="submit" value="LOGIN" />' +
                                '</form>' +
                                '<div style="width: 100%;
height:30px; margin-left:42%;">' +
                                '<p><font color="red">Incorrect user
data!</font></p>' +
                                '</div>' +
                                '</div>' +
                                '</body>' +
                                '</html>';

                    response.writeHead(200, { "Content-Type":
"text/html" });
                    response.write(body);
                }
            });

        }).on("error", (err) => {
            console.log("Error: " + err.message);
            userID = 'false';
            console.log("Verification user status2: " + userID);
```

```
        var body = '<html>' +
                            '<head>' +
                            '<script type="text/javascript">' +
                            'function(error){' +
                                  'if(error.responseText ==
"showAlert")' +
                                        'alert("ALAAAAAARM!! Please
enter correct user name and password.");' +
                                        '</script>' +
                            '<meta http-equiv="Content-Type"
content="text/html; ' +
                            'charset=UTF-8" />' +
                            '</head>' +
                            '<body>' + //border-bottom: 3px double
gray;
                            '<div style="width: 100%; height:30px;
top:0; margin-top: 24px; margin-left:48%; padding-bottom:
10px;">' +
                            '<p>RED DROP BOX<img src="logo.png"
width="30px" height="30px"></p>' +
                            '</div>' +
                            '<div style=" border-bottom: 3px
double gray; width: 90%; height:1px; margin-left:5%"/>' +
                            '<div style=" width: 150px;
height:300px; position: absolute; top:0; bottom: 0; left: 0;
right: 0;  margin: auto;">' +
                            '<form action="/loginIn"
method="post">' +
                            '<p>Login:</p><input type="text"
name="login" rows="1" cols="20"/><br>' +
                            '<p>Password:</p><input
type="password" name="pass" rows="1" cols="20"/><br>' +
                            '<br><input style="margin-left:60px"
type="submit" value="LOGIN" />' +
                            '</form>' +
                            '<div style="width: 100%; height:30px;
margin-left:42%;">' +
                            '<p><font color="red">ERROR:
verification server is unavalible.\nPlease, try again
latter.</font></p>' +
                            '</div>' +
                            '</div>' +
                            '</body>' +
                            '</html>';

        response.writeHead(200, { "Content-Type": "text/html"
});
        response.write(body);
    });
}
```

```
exports.checkStatus = checkStatus;
     exports.userVer = userVer;
```

Файл: requestHandlers.js

```
var querystring = require("querystring");
var verApi = require("./verApi");
var fs = require('fs');
var http = require('http');
var formidable = require('formidable');
var url = require("url");

var BalanceService = "http://localhost";
var BalanceService_Port = 7000;

var DateTimeService = "http://localhost";
var DateTimeService_Port = 7001;

function start(response, postData) {
    console.log("Request handler 'start' was called.");

    var body = '<html>' +
        '<head>' +
        '<script type="text/javascript">' +
        'function(error){' +
        'if(error.responseText == "showAlert")' +
        'alert("ALAAAAAARM!! Please enter correct user name and
password.");' +
        '</script>' +
        '<meta http-equiv="Content-Type" content="text/html; ' +
        'charset=UTF-8" />' +
        '</head>' +
        '<body>' + //border-bottom: 3px double gray;
        '<div style="width: 100%; height:30px; top:0; margin-
top: 24px; margin-left:48%; padding-bottom: 10px;">' +
        '<p>RED DROP BOX<img src="logo.png" width="30px"
height="30px"></p>' +
        '</div>' +
        '<div style=" border-bottom: 3px double gray; width:
90%; height:1px; margin-left:5%"/>' +
        '<div style=" width: 150px; height:300px; position:
absolute; top:0; bottom: 0; left: 0; right: 0;  margin: auto;">'
+
        '<form action="/loginIn" method="post">' +
        '<p>Login:</p><input type="text" name="login" rows="1"
cols="20"/><br>' +
        '<p>Password:</p><input type="password" name="pass"
rows="1" cols="20"/><br>' +
        '<br><input style="margin-left:60px" type="submit"
value="LOGIN" />' +
        '</form>' +
        '</div>' +
        '</body>' +
```

```
        '</html>';

    response.writeHead(200, { "Content-Type": "text/html" });
    response.write(body);
    response.end();
}

function loginIn(response, postData) {
    console.log("Request handler 'loginIn5555' was called.");

    console.log("postData: " + postData);

    response.writeHead(200, { "Content-Type": "text/plain" });

    var login = postData.split('&')[0].split('=')[1];
    var pass = postData.split('&')[1].split('=')[1];

    verApi.userVer(login, pass, response);
}

function newAdvice(response, postData, request) {
    console.log("Request handler 'newAdvice' was called.");
    console.log("postData: " + postData);
    console.log("request.url: " + request.url);

    var userName =
url.parse(request.url).search.split("user=")[1].split('&')[0];
    var userID =
url.parse(request.url).search.split("userid=")[1];

    http.get(BalanceService + ':' + BalanceService_Port +
'/getUserInfo?userID=' + userID, (resp) => {
        let data = '';

        resp.on('data', (chunk) => {
            data += chunk;
        });

        resp.on('end', () => {

            console.log("AllData00000000000000000000: " + data);
            allData = data;

            var userRaiting = parseFloat(allData.split(' ')[2]);
            var advicePrice = 0;

            console.log("userRaiting1: " + userRaiting);

            if(userRaiting >= 10.0)
            {
                advicePrice = userRaiting * (10 * (userRaiting -
10) / (100 - userRaiting + 1));
```

```
                console.log("advicePrice1: " + advicePrice);

                advicePrice = parseInt(advicePrice);
                console.log("advicePrice2: " + advicePrice);
            }

            var folderPath = __dirname + '\\newadvice.html';
            var body = fs.readFileSync(folderPath).toString();

            body = body.replace("%username%", userName);
            body = body.replace("%adviceprice%", advicePrice);

            body = body.split("%params%").join("user=" +
userName + "&userid=" + userID);

            response.writeHead(200, { "Content-Type":
"text/html" });
            response.write(body);
            response.end();

        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
    });
}

function saveAdvice(response, postData, request) {
    console.log("Request handler 'saveAdvice' was called.");
    console.log("postData: " + postData);
    console.log("request.url: " + request.url);

    var userName =
url.parse(request.url).search.split("user=")[1].split('&')[0];
    var userID =
url.parse(request.url).search.split("userid=")[1];

    http.get(BalanceService + ':' + BalanceService_Port +
'/saveadvice?userid=' + userID + '&' + postData, (resp) => {
        let data = '';

        // A chunk of data has been recieved.
        resp.on('data', (chunk) => {
            data += chunk;
        });

        // The whole response has been received. Print out the
result.
        resp.on('end', () => {

            console.log("AllData: " + data);
            allData = data;
```

```
            response.writeHead(302, {
                'Location': '/advices?user=' + userName +
'&userid=' + userID
                //add other headers here...
            });
            response.end();
        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
    });
}

function buyAdvice(response, postData, request) {
    console.log("Request handler 'buyAdvice' was called.");
    console.log("postData: " + postData);
    console.log("request.url: " + request.url);

    var userName =
url.parse(request.url).search.split("user=")[1].split('&')[0];
    var userID =
url.parse(request.url).search.split("userid=")[1].split('&')[0];

    http.get(BalanceService + ':' + BalanceService_Port +
request.url, (resp) => {
        let data = '';

        // A chunk of data has been recieved.
        resp.on('data', (chunk) => {
            data += chunk;
        });

        // The whole response has been received. Print out the
result.
        resp.on('end', () => {

            console.log("AllData: " + data);
            allData = data;

            if (allData == '')
            {
                response.writeHead(302,
                    {
                    'Location': '/advices?user=' + userName +
'&userid=' + userID
                    //add other headers here...
                });
            }
            else
            {
                response.writeHead(302,
```

```
                    {
                        'Location': '/advices?user=' + userName
+ '&userid=' + userID + '&errid=1'
                        //add other headers here...
                    });
            }
            response.end();
        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
    });
}

function getAdvice(response, postData, request) {
    console.log("Request handler 'getAdvice' was called.");
    console.log("postData: " + postData);
    console.log("request.url: " + request.url);

    var userName =
url.parse(request.url).search.split("user=")[1].split('&')[0];
    var userID =
url.parse(request.url).search.split("userid=")[1].split('&')[0];
    var adviceID =
url.parse(request.url).search.split("adviceid=")[1];

    http.get(BalanceService + ':' + BalanceService_Port +
request.url, (resp) => {
        let data = '';

        // A chunk of data has been recieved.
        resp.on('data', (chunk) => {
            data += chunk;
        });

        // The whole response has been received. Print out the
result.
        resp.on('end', () => {

            console.log("AllData222222220: " + data);
            allData = data.split(' ');

            var folderPath = __dirname + '\\getadvice.html';
            var body = fs.readFileSync(folderPath).toString();

            body = body.replace("%expert%", allData[0]);
            body = body.replace("%karma%", allData[1]);
            body = body.replace("%item%", allData[2]);
            body = body.replace("%datestart%", allData[3]);
            body = body.replace("%dateend%", allData[4] + " (" +
allData[5] + " d.)");
            body = body.replace("%startprice%", allData[6]);
```

```
            body = body.replace("%takeprofit%", allData[7]);
            body = body.replace("%stoploss%", allData[8]);
            body = body.replace("%plus%", allData[9]);
            body = body.replace("%minus%", allData[10]);

            var adviceStatusCode = parseInt(allData[12]);
            var infoString = "";

            console.log("adviceStatusCode: " +
adviceStatusCode);

            if (adviceStatusCode == 1)
            {
                body = body.replace("%status%", "Активен");
            }
            else
            {
                body = body.replace("%status%", "Завершен");

                var reasonCode = parseInt(allData[13]);
                var reasonString = '<tr><td width="100px"
style=""><p style="margin-left:50px ;text-align: left; font-
size:medium;">Причина:</p></td><td width="100px" style="text-
align:left;">';
                if (reasonCode == 1)
                {
                    reasonString += "достигнут TakeProfit";
                }
                else
                {
                    if (reasonCode == 2)
                    {
                        reasonString += "достигнут StopLoss";
                    }
                    else
                    {
                        reasonString += "достигнут конец
периода";
                    }
                }

                reasonString += '</td><td width="50px"
style="text-align:center;"></td><td width="100px" style="text-
align:center;"></td></tr>'

                var profit = parseInt(allData[14]);
                var profitString = "";

                if (profit < 0)
                {
                    profitString = '<tr><td width="100px"
style=""><p style="margin-left:50px ;text-align: left; font-
```

```
size:medium;">Потеря:</p></td><td width="100px" style="text-
align:left;">' + profit * (-1) + '%';
                }
                else
                {
                    profitString = '<tr><td width="100px"
style=""><p style="margin-left:50px ;text-align: left; font-
size:medium;">Прибыль:</p></td><td width="100px" style="text-
align:left;">' + profit + '%';
                }

                profitString += '</td><td width="50px"
style="text-align:center;"></td><td width="100px" style="text-
align:center;"></td></tr>';

                infoString = reasonString + "</br>" +
profitString;
            }

            body = body.replace("%info%", infoString);
            body = body.replace("%params%", 'user=' + userName +
'&userid=' + userID);

            response.writeHead(200, { "Content-Type":
"text/html" });
            response.write(body);
            response.end();
        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
    });
}

function mysessions(response, postData, request) {
    console.log("Request handler 'mysessions' was called.");

    console.log("postData: " + postData);
    console.log("request.url: " + request.url);

    var userName =
url.parse(request.url).search.split("user=")[1].split('&')[0];
    var userID =
url.parse(request.url).search.split("userid=")[1];

    var folderPath = __dirname + '\\my_sessions.html';
    var body = fs.readFileSync(folderPath).toString();

    http.get(BalanceService + ':' + BalanceService_Port +
'/getUserInfo?user=' + userName, (resp) => {
        let data = '';
```

```
        // A chunk of data has been recieved.
        resp.on('data', (chunk) => {
            data += chunk;
        });

        // The whole response has been received. Print out the
result.
        resp.on('end', () => {

            console.log("AllData777777777: " + data);
            allData = data;

        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
    });
}

function advices(response, postData, request) {
    console.log("Request handler 'advices' was called.");

    console.log("search: " + url.parse(request.url).search);

    var userName =
url.parse(request.url).search.split("user=")[1].split('&')[0];
    var userID = '';
    var errID = '';

    console.log("params length:" +
url.parse(request.url).search.split("&").length);

    for (var i = 0; i <
url.parse(request.url).search.split("&").length; i++)
    {
        console.log("value: " +
url.parse(request.url).search.split("&")[i]);
    }

    if(url.parse(request.url).search.split("&").length < 3)
    {
        userID =
url.parse(request.url).search.split("userid=")[1];
    }
    else
    {
        userID =
url.parse(request.url).search.split("userid=")[1].split('&')[0];
        errID =
url.parse(request.url).search.split("errid=")[1];
    }
```

```
    console.log("userID: " + userID);
    console.log("errID: " + errID);

    var folderPath = __dirname + '\\advices.html';
    var body = fs.readFileSync(folderPath).toString();

    body = body.replace("%username%", userName);
    body = body.replace("%username2%", userName);
    body = body.split("%params%").join("user=" + userName +
"&userid=" + userID);

    http.get(BalanceService + ':' + BalanceService_Port +
'/getUserInfo?userID=' + userID, (resp) => {
        let data = '';

        // A chunk of data has been recieved.
        resp.on('data', (chunk) => {
            data += chunk;
        });

        // The whole response has been received. Print out the
result.
        resp.on('end', () => {

            console.log("AllData777777777: " + data);
            allData = data;

            if (allData != null && allData != undefined &&
allData != "")
            {
                body = body.replace("%balance%", allData.split('
')[1] + " $");
                body = body.replace("%karma%", allData.split('
')[2]);
                body = body.replace("%params%", userName);
                //body = body.replace("%testcheckdata%", '<a
href="' + BalanceService + ':' + BalanceService_Port +
'/testcheckdata">TEST</a>');
            }

            //Текущая дата
            http.get(DateTimeService + ':' +
DateTimeService_Port + '/getcurrentdate', (resp) => {
                let data = '';

                resp.on('data', (chunk) => {
                    data += chunk;
                });

                resp.on('end', () => {
                    var dateTime = new
Date(parseInt(parseInt(data * 1)));
```

```
                    console.log("dateTime: " +
parseInt(dateTime*1));
                    console.log("dateTime: " +
dateTime.getDate() + "." + dateTime.getMonth() + "." +
dateTime.getFullYear());

                    var dateTime_new = "";
                    if (parseInt(dateTime.getDate()) < 10)
                    {
                        dateTime_new += "0" +
dateTime.getDate();
                    }
                    else
                    {
                        dateTime_new += dateTime.getDate();
                    }

                    dateTime_new += ".";

                    if (parseInt(dateTime.getMonth() + 1) < 10)
                    {
                        dateTime_new += "0" +
parseInt(dateTime.getMonth() + 1);
                    }
                    else
                    {
                        dateTime_new +=
parseInt(dateTime.getMonth() + 1);
                    }

                    dateTime_new += ".";
                    dateTime_new += dateTime.getFullYear();


                    body = body.replace("%datetime%",
dateTime_new);

                    //Советы экспертов
                    http.get(BalanceService + ':' +
BalanceService_Port + '/getAdvices', (resp) => {
                        let data = '';

                        resp.on('data', (chunk) => {
                            data += chunk;
                        });

                        resp.on('end', () => {

                            console.log("AllData111: " + data);
                            allData = data;
```

```
                                var advicesHTML = '<table
style="width: 90%; height:100%; margin-left: 5%; text-align:
left; font-size:medium;"><tr style="height: 10%"><td
style="text-align: center; border-top: 2px solid Silver; border-
bottom: 2px solid Silver">Эксперт</td><td style="text-align:
center; border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Рейтинг эксперта</td><td style="text-align: center;
border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Начало сессии</td><td style="text-align: center; border-
top: 2px solid Silver; border-bottom: 2px solid
Silver">Окончание сессии</td><td style="text-align: center;
border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Продолжительность сессии (дней)</td><td style="text-
align: center; border-top: 2px solid Silver; border-bottom: 2px
solid Silver">Ожидаемая прибыль (%)</td><td style="text-align:
center; border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Возможная потеря (%)</td><td style="text-align: center;
border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Стоимость совета</td><td style="text-align: center;
border: none"></td></tr>';

                                if (allData != null && allData !=
undefined && allData != "") {
                                    var allAdvicesRow =
allData.split('\r\n');

                                    for (var i = 0; i <
allAdvicesRow.length; i++) {
                                        if (allAdvicesRow[i] != null
&& allAdvicesRow[i] != undefined) {
                                            if (i % 2 == 0) {
                                                advicesHTML += '<tr
onclick="" style="background: rgba(180, 180, 180, 0.2); text-
align: center;"><td style="height: 30px;" id="expertName">' +
allAdvicesRow[i].split(' ')[2] + '</td><td>' +
allAdvicesRow[i].split(' ')[1] + '</td><td>' +
allAdvicesRow[i].split(' ')[3] + '</td><td>' +
allAdvicesRow[i].split(' ')[4] + '</td><td>' +
allAdvicesRow[i].split(' ')[5] + '</td><td>' +
(Math.round(allAdvicesRow[i].split(' ')[6] * 100) / 100) +
'</td><td>' + (Math.round(allAdvicesRow[i].split(' ')[7] * 100)
/ 100) + '</td><td>' + allAdvicesRow[i].split(' ')[8] +
'</td><td>' + '<a href="/buyadvice?user=' + userName +
'&userid=' + userID + '&adviceid=' + allAdvicesRow[i].split('
')[0] + '"' + ' style="border: 1px solid black;"><font
style="color:black; size: small;">Приобрести</font></a>' +
'</td></tr>';
                                            }
                                            else {
                                                advicesHTML += '<tr
onclick="" style="background: rgba(200, 200, 200, 0.2); text-
align: center;"><td style="height: 30px;" id="expertName">' +
```

```
allAdvicesRow[i].split(' ')[2] + '</td><td>' +
allAdvicesRow[i].split(' ')[1] + '</td><td>' +
allAdvicesRow[i].split(' ')[3] + '</td><td>' +
allAdvicesRow[i].split(' ')[4] + '</td><td>' +
allAdvicesRow[i].split(' ')[5] + '</td><td>' +
(Math.round(allAdvicesRow[i].split(' ')[6] * 100) / 100) +
'</td><td>' + (Math.round(allAdvicesRow[i].split(' ')[7] * 100)
/ 100) + '</td><td>' + allAdvicesRow[i].split(' ')[8] +
'</td><td>' + '<a href="/buyadvice?user=' + userName +
'&userid=' + userID + '&adviceid=' + allAdvicesRow[i].split('
')[0] + '"' + ' style="border: 1px solid black;"><font
style="color:black; size: small;">Приобрести</font></a>' +
'</td></tr>';
                                        }
                                    }
                                    else {
                                        if (i ==
allAdvicesRow.length) {
                                            advicesHTML += '<tr
style="height: 10%"><td></td></tr>';
                                        }
                                        else {
                                            advicesHTML += '<tr
style="height: 10%"><td></td></tr>';
                                        }
                                    }
                                }
                            }

                            advicesHTML += '<tr><td colspan="6"
style="text-align: left;"><a href="/newadvice?user=' + userName
+ '&userid=' + userID + '" style="text-decoration:none"><font
color="black"><b>Новый совет
эксперта..</b></font></a></td></tr>';
                            advicesHTML += '</table>';
                            body = body.replace("%content%",
advicesHTML);

                            allData = null;

                            //Приобретенные советы
                            http.get(BalanceService + ':' +
BalanceService_Port + '/getBoughtAdvices?userID=' + userID,
(resp) => {
                                let data = '';

                                resp.on('data', (chunk) => {
                                    data += chunk;
                                });

                                resp.on('end', () => {
```

```
                                    console.log("AllData222: " +
data);
                                    allData = data;

                                    advicesHTML = '<table
style="width: 90%; height:100%; margin-left: 5%; text-align:
left; font-size:medium;"><tr style="height: 10%"><td
style="text-align: center; border-top: 2px solid Silver; border-
bottom: 2px solid Silver">Эксперт</td><td style="text-align:
center; border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Рейтинг эксперта</td><td style="text-align: center;
border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Начало сессии</td><td style="text-align: center; border-
top: 2px solid Silver; border-bottom: 2px solid
Silver">Окончание сессии</td><td style="text-align: center;
border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Продолжительность сессии</td><td style="text-align:
center; border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Ожидаемая прибыль (%)</td><td style="text-align: center;
border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Возможная потеря (%)</td><td style="text-align: center;
border-top: 2px solid Silver; border-bottom: 2px solid
Silver">Стоимость совета</td><td style="text-align: center;
border:none"></td>';

                                    if (allData != null &&
allData != undefined && allData != "") {

                                        var allDataParts =
allData.split('\r\n');

                                        for (var i = 0; i <
allDataParts.length; i++) {
                                            if (allDataParts[i]
!= null && allDataParts[i] != undefined) {
                                                if (i % 2 == 0)
{
                                                    advicesHTML
+= '<tr onclick="" style=" background: rgba(180, 180, 180, 0.2);
text-align: center;"><td style="height: 30px;"
id="expertName">' + allDataParts[i].split(' ')[2] + '</td><td>'
+ allDataParts[i].split(' ')[1] + '</td><td>' +
allDataParts[i].split(' ')[3] + '</td><td>' +
allDataParts[i].split(' ')[4] + '</td><td>' +
allDataParts[i].split(' ')[5] + '</td><td>' +
(Math.round(allDataParts[i].split(' ')[6] * 100) / 100) +
'</td><td>' + (Math.round(allDataParts[i].split(' ')[7] * 100) /
100) + '</td><td>' + allDataParts[i].split(' ')[8] + '</td><td>'
+ '<a href="/getadvice?user=' + userName + '&userid=' + userID +
'&adviceid=' + allDataParts[i].split(' ')[0] + '"' + '
style="border: 1px solid black;"><font style="color:black; size:
small;">Подробнее</font></a>' + '</td></tr>';
```

```
                                                }
                                                else {
                                                    advicesHTML
+= '<tr onclick="" style=" background: rgba(200, 200, 200, 0.2);
text-align: center;"><td style="height: 30px;"
id="expertName">' + allDataParts[i].split(' ')[2] + '</td><td>'
+ allDataParts[i].split(' ')[1] + '</td><td>' +
allDataParts[i].split(' ')[3] + '</td><td>' +
allDataParts[i].split(' ')[4] + '</td><td>' +
allDataParts[i].split(' ')[5] + '</td><td>' +
(Math.round(allDataParts[i].split(' ')[6] * 100) / 100) +
'</td><td>' + (Math.round(allDataParts[i].split(' ')[7] * 100) /
100) + '</td><td>' + allDataParts[i].split(' ')[8] + '</td><td>'
+ '<a href="/getadvice?user=' + userName + '&userid=' + userID +
'&adviceid=' + allDataParts[i].split(' ')[0] + '"' + '
style="border: 1px solid black;"><font style="color:black; size:
small;">Подробнее</font></a>' + '</td></tr>';
                                                }
                                            }
                                            else {
                                                if (i ==
allDataParts.length) {
                                                    advicesHTML
+= '<tr style="height: 10%"><td></td></tr>';
                                                }
                                                else {
                                                    advicesHTML
+= '<tr style="height: 10%"><td></td></tr>';
                                                }
                                            }
                                        }
                                    }

                                    advicesHTML += '<tr><td
colspan="6" style="text-align: left;"></td></tr>';
                                    advicesHTML += '</table>';
                                    body =
body.replace("%mycontent%", advicesHTML);

                                    response.writeHead(200, {
"Content-Type": "text/html" });
                                    response.write(body);
                                    response.end();
                                });

                            }).on("error", (err) => {
                                console.log("Error: " +
err.message);
                            });
                        });

                    }).on("error", (err) => {
```

```
                        console.log("Error: " + err.message);
                    });
                });

            }).on("error", (err) => {
                console.log("Error: " + err.message);
            });
        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
    });
}

function upload(response, postData, request) {
    console.log("UPLOADrequest: " + request.url);

    var newRequestUrl = BalanceService + ':' +
BalanceService_Port + '/getDataServer' +
url.parse(request.url).search + '&action=' +
url.parse(request.url).pathname.split('/')[1];

    console.log("newRequestUrl: " + newRequestUrl);

    http.get(newRequestUrl, (resp1) => {
        let dataServerUrl = '';

        resp1.on('data', (chunk) => {
            dataServerUrl += chunk;
        });

        resp1.on('end', () => {
            console.log("Balance Response: " + dataServerUrl);
            var body = '<html>' +
                '<head>' +
                '<script type="text/javascript">' +
                'function addUser() {
' +
                'document.getElementById("postForm").action =
document.getElementById("postForm").action +
document.getElementById("userNameDiv").innerHTML;' +

'alert(document.getElementById("postForm").action);' +
                'document.getElementById("postForm").submit();'
+
                '</script>' +
                '<meta http-equiv="Content-Type"
content="text/html; ' +
                'charset=UTF-8" />' +
                '</head>' +
                '<body>' + //border-bottom: 3px double gray;
```

```
                '<div style="width: 100%; height:30px; top:0;
margin-top: 24px; margin-left:48%; padding-bottom: 10px;">' +
                '<p>RED DROP BOX<img src="logo.png" width="30px"
height="30px"></p>' +
                '</div>' +
                '<div style=" border-bottom: 3px double gray;
width: 90%; height:1px; margin-left:5%"/>' +
                '<div style=" width: 150px; height:300px;
position: absolute; top:0; bottom: 0; left: 0; right: 0;
margin: auto;">' +
                '<form action="' + dataServerUrl +
url.parse(request.url).path + '" method="post"
enctype="multipart/form-data" id="postForm">' +
                '<input type="file" name="upfile">' +
                '<a href="#"
onclick="document.getElementById(\'postForm\').submit();">
Upload </a>' +
                '</form>' +
                '</div>' +
                '</body></html>';

            response.writeHead(200, { "Content-Type":
"text/html" });
            response.write(body);
            response.end();



        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
    });
}

function load(response, postData, request) {
    console.log("request: " + request.url);

    var newRequestUrl = BalanceService + ':' +
BalanceService_Port + '/getDataServer' +
url.parse(request.url).search + '&action=' +
url.parse(request.url).pathname.split('/')[1];

    console.log("newRequestUrl: " + newRequestUrl);

    http.get(newRequestUrl, (resp1) => {
        let dataServerUrl = '';

        resp1.on('data', (chunk) => {
            dataServerUrl += chunk;
        });
```

```
        resp1.on('end', () => {

            var body = '<html>' +
                '<head>' +
                '<meta http-equiv="Content-Type"
content="text/html; ' +
                'charset=UTF-8" />' +
                '</head>' +
                '<body>' + //border-bottom: 3px double gray;
                '<div style="width: 100%; height:30px; top:0;
margin-top: 24px; margin-left:48%; padding-bottom: 10px;">' +
                '<p>RED DROP BOX<img src="logo.png" width="30px"
height="30px"></p>' +
                '</div>' +
                '<div style=" border-bottom: 3px double gray;
width: 90%; height:1px; margin-left:5%"/>' +
                '<div style=" width: 150px; height:300px;
position: absolute; top:0; bottom: 0; left: 0; right: 0;
margin: auto;">' +
                '<br<br><a href="' + dataServerUrl +
url.parse(request.url).path + '">Download</a>' +
                '</div>' +
                '</body>' +
                '</html>';

            response.writeHead(200, { "Content-Type":
"text/html" });
            response.end(body);
        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
    });
}

function del(response, postData, request) {

    console.log("request: " + request.url);

    var newRequestUrl = BalanceService + ':' +
BalanceService_Port + '/getDataServer' +
url.parse(request.url).search + '&action=' +
url.parse(request.url).pathname.split('/')[1];

    console.log("newRequestUrl: " + newRequestUrl);

    http.get(newRequestUrl, (resp1) => {
        let dataServerUrl = '';

        resp1.on('data', (chunk) => {
            dataServerUrl += chunk;
        });
```

```
        resp1.on('end', () => {
            console.log("Url for Response: " + dataServerUrl +
url.parse(request.url).path);

            http.get(dataServerUrl +
url.parse(request.url).path, (resp) => {
                let data = '';

                // A chunk of data has been recieved.
                resp.on('data', (chunk) => {
                    data += chunk;
                });

                // The whole response has been received. Print
out the result.
                resp.on('end', () => {
                    //console.log(JSON.parse(data).explanation);
                    console.log(data);

                    response.writeHead(200, { "Content-Type":
"text/html" });
                    response.write(data);
                    response.end();
                });

            }).on("error", (err) => {
                console.log("Error: " + err.message);
            });

        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
    });
}

function registration(response, postData) {
    var body = '<html>' +
        '<head>' +
        '<meta http-equiv="Content-Type" content="text/html; ' +
        'charset=UTF-8" />' +
        '</head>' +
        '<body>' + //border-bottom: 3px double gray;
        '<div style="width: 100%; height:30px; top:0; margin-
top: 24px; margin-left:48%; padding-bottom: 10px;">' +
        '<p>J I T S W E R</p>' +
        '</div>' +
        '<div style=" border-bottom: 3px double gray; width:
90%; height:1px; margin-left:5%"/>' +
```

```
        '<div style=" width: 150px; height:300px; position:
absolute; top:0; bottom: 0; left: 0; right: 0;  margin: auto;">'
+
        '<form action="/upload" method="post">' +
        '<p>New login:</p><input type="text" name="login"
rows="1" cols="20"/><br>' +
        '<p>Password:</p><input type="password" name="pass"
rows="1" cols="20"/><br>' +
        '<br><input style="margin-left:60px" type="submit"
value="REG!" />' +
        '</form>' +
        '</div>' +
        '</body>' +
        '</html>';

    response.writeHead(200, { "Content-Type": "text/html" });
    response.write(body);
    response.end();
}

function getLogo(response, postData) {
    var img = fs.readFileSync('./logo.png');
    response.writeHead(200, { 'Content-Type': 'image/png' });
    response.end(img, 'binary');
}

function getUserIcon(response, postData) {
    var img = fs.readFileSync('./icon_user.png');
    response.writeHead(200, { 'Content-Type': 'image/png' });
    response.end(img, 'binary');
}

function getBcg(response, postData) {
    var img = fs.readFileSync('./bcg2.png');
    response.writeHead(200, { 'Content-Type': 'image/png' });
    response.end(img, 'binary');
}

function fileUpload(response, postData) {
    response.write(postData);
    response.end();
}

exports.start = start;
exports.loginIn = loginIn;
exports.mysessions = mysessions;
exports.advices = advices;
exports.newAdvice = newAdvice;
exports.saveAdvice = saveAdvice;
exports.buyAdvice = buyAdvice;
exports.getAdvice = getAdvice;
exports.upload = upload;
```

```
exports.load = load;
exports.del = del;
exports.registration = registration;
exports.getLogo = getLogo;
exports.getUserIcon = getUserIcon;
exports.getBcg = getBcg;
exports.fileUpload = fileUpload;
```

Модуль: Сервис балансировки нагрузки

Файл: server.js

```
exports.fileUpload = fileUpload;
sdfds
var http = require("http");
var url = require("url");


var express = require("express");
var upload = require("express-fileupload");
var app = express();
app.use(upload());

var fs = require('fs');

var mkdirp = require('mkdirp');

var DataService_1 = "http://192.168.43.245";
var DataService_1_Port = 7771;

var DataService_2 = "http://192.168.43.245";
var DataService_2_Port = 7772;

var serverIs = true;

function getEffictiveService(req)
{
    var DataServiceUrl;
    var DataServicePort;

    if (serverIs) {
        DataServiceUrl = DataService_1;
        DataServicePort = DataService_1_Port;
        serverIs = !serverIs;
    }
```

```
    else {
        DataServiceUrl = DataService_2;
        DataServicePort = DataService_2_Port;
        serverIs = !serverIs;
    }

    return DataServiceUrl + ':' + DataServicePort;
}

function Replication() {
    console.log(' === Replication started === ');

    var folderPath = __dirname + '/uploads/';

    var usersdirs = '';

    fs.readdirSync(folderPath).forEach(file => {
        if (usersdirs != '') {
            usersdirs += '|';
        }

        usersdirs += file;
    })

    if (usersdirs != null && usersdirs != undefined && usersdirs
!= '') {

        usersdirs = usersdirs.split('|');

        usersdirs.forEach(user => {
            console.log('');
            console.log('user: ' + user);

            var userdir = folderPath + user;

            var allFiles = fs.readFileSync(userdir +
'\\AllFiles.rdb').toString();

            var replicationData = fs.readFileSync(userdir +
'\\Transactions.rpl').toString();
            console.log('replicationData: ' + replicationData);

            var transactions = replicationData.split(';');

            transactions.forEach(function (transaction, i, arr)
{
                if (i != (arr.length - 1)) {
                    var file = transaction.split(':')[0];
                    var action =
transaction.split(':')[1].split('_')[0];
                    var dsidsource =
transaction.split(':')[1].split('_')[1];
```

```
                    var dsiddest;

                    if (dsidsource == 1) {
                        dsiddest = 2;
                    }
                    else {
                        dsiddest = 1;
                    }

                    console.log('');
                    console.log('file: ' + file);
                    console.log('action: ' + action);
                    console.log('dsidsource: ' + dsidsource);
                    console.log('dsiddest: ' + dsiddest);

                    console.log('');
                    console.log('make replication');

                    var requesturl;

                    if (dsiddest == 1) {
                        requesturl = DataService_1 + ':' +
DataService_1_Port;
                    }
                    else {
                        requesturl = DataService_2 + ':' +
DataService_2_Port;
                    }

                    if (action == 'add')
                    {
                        requesturl += '/repladd?user=' + user +
'&file=' + file;
                    }
                    else
                    {
                        requesturl += '/repldel?user=' + user +
'&file=' + file;
                    }

                    console.log('requesturl: ' + requesturl);

                    http.get(requesturl, (resp2) => {

                        let data = '';

                        resp2.on('data', (chunk) => {

                        });

                        resp2.on('end', () => {
```

```
                            //console.log("EffictiveService is "
+ requestUrl);

                            //res.writeHead(200, { "Content-
Type": "text/html" });
                            //res.write(requestUrl);
                            //res.end();
                        });

                    }).on("error", (err) => {
                        console.log("Error: " + err.message);
                    });
                }
            });

            transactions.forEach(function (transaction, i, arr)
{
                if (i != (arr.length - 1)) {

                    var file = transaction.split(':')[0];
                    var action =
transaction.split(':')[1].split('_')[0];
                    var dsidsource =
transaction.split(':')[1].split('_')[1];
                    var dsiddest;

                    if (dsidsource == 1) {
                        dsiddest = 2;
                    }
                    else {
                        dsiddest = 1;
                    }

                    if (action == 'add') {
                        allFiles = allFiles.split(file + ':' +
dsidsource + ';');
                        allFiles = allFiles[0] + file + ':12;' +
allFiles[1];
                    }
                    else
                    {
                        //allFiles = allFiles.split(file + ':' +
dsidsource + ';');
                        //allFiles = allFiles[0] + allFiles[1];
                    }
                }
            });

            //Все транзакции реплицировали. Очистим список
транзакций.
            fs.writeFile(userdir + '\\Transactions.rpl', '',
function (err) {
```

```
                if (err) throw err;
            });

            console.log("AllFiles: " + allFiles);

            //Теперь приведем список всех файлов пользователя в
действительность, указав, что файл доступен с нескольких
серверов
            fs.writeFile(userdir + '\\AllFiles.rdb', allFiles,
function (err) {
                if (err) throw err;
            });

        })
    }

    console.log(' === Replication END === ');
    setTimeout(Replication, 20000);
}

app.get('/getDataServer', function (req, res) {
    try
    {
        var folderPath = __dirname + '/uploads/';

        console.log('req.url1: ' + req.url);

        var username =
url.parse(req.url).search.split("user=")[1].split("&")[0];
        console.log("username1: " + username);

        var action =
decodeURIComponent(url.parse(req.url).search.split("action=")[1]
.split("&")[0]);
        console.log("action1: " + action);

        var fileName = '';

        if (action == 'load' || action == 'delete')
        {
            fileName =
decodeURIComponent(url.parse(req.url).search.split("file=")[1].s
plit("&")[0]);
            console.log("fileName1: " + fileName);
        }

        var userdir = folderPath + username + '/';

        var requestUrl = '';

        //Учтены только те случаи, которые не отсечены логикой
приложения в других модулях
```

```
        //считываю список всех файлов юзера

        var allFiles = fs.readFileSync(userdir +
'AllFiles.rdb').toString();

        var dsID = '';
        var dsIDTemp = '';

        if (action == 'load' || action == 'delete')
        {
            dsIDTemp = allFiles.split(fileName + ':')[1];

            if (dsIDTemp.length != 1) {
                dsID = dsIDTemp.split(';')[0];
            }
        }

        if (action == 'load')
        {
            //файл точно существует, потому что его скачиваем
            //возможны два варианта:
            //2.Лежит на обоих сервах и одинаковый везде
            //получаем приоритетный серв - возвращаем его.
            //3.Лежит на одном серве, еще не успел
среплицироваться на второй - возвращаем тот, где есть.

            if(dsID == 12)
            {
                requestUrl = getEffictiveService(req);
            }
            else
            {
                if (dsID == 1) {
                    requestUrl = DataService_1 + ':' +
DataService_1_Port;
                }
                else
                {
                    requestUrl = DataService_2 + ':' +
DataService_2_Port;
                }
            }
            console.log('action is load request: ' +
requestUrl);
        }
        else
        {
            if (action == 'upload')
            {
                //костыль
                requestUrl = getEffictiveService(req);
```

```
                console.log('action is upload request: ' +
requestUrl);
            }
            else
            {
                if (action == 'delete')
                {
                    console.log('action is del');
                    //файл точно существует, потому что его
удаляем
                    //смотрю, на каких сервах лежит файл сейчас
                    //удаляю строку
                    //если на обоих сервах - просто получаю
приоритетный удаляю там, остальное сделает репликация.
                    //если на одном - то возможна только одна
ситуация:
                    //файл залили, а среплицироваться он еще не
успел
                    //проверяем список транзакций для репликации
                    //убираем все транзакции с этим файлом
                    //удаляю файл

                    if (dsID == 12)
                    {
                        requestUrl = getEffictiveService(req);
                    }
                    else
                    {
                        if (dsID == 1) {
                            requestUrl = DataService_1 + ':' +
DataService_1_Port;
                        }
                        else {
                            requestUrl = DataService_2 + ':' +
DataService_2_Port;
                        }
                    }

                    console.log('action is del request: ' +
requestUrl);

                }
            }
        }

        console.log("req for ping: " +
requestUrl.split("?")[0]);

        http.get(requestUrl.split("?")[0], (resp1) => {

            let data = '';
```

```
            resp1.on('data', (chunk) => {

            });

            resp1.on('end', () => {
                console.log("data: " + data);

                console.log("EffictiveService is " +
requestUrl);

                res.writeHead(200, { "Content-Type": "text/html"
});
                res.write(requestUrl);
                res.end();
            });

        }).on("error", (err) => {
            console.log("Error: " + err.message);

            var requestUrl = getEffictiveService(req);

            http.get(requestUrl.split("?")[0], (resp2) => {

                let data = '';

                resp2.on('data', (chunk) => {

                });

                resp2.on('end', () => {
                    console.log("data: " + data);

                    console.log("EffictiveService is " +
requestUrl);

                    res.writeHead(200, { "Content-Type":
"text/html" });
                    res.write(requestUrl);
                    res.end();
                });

            }).on("error", (err) => {
                console.log("Error: " + err.message);
            });

        });
    }
    catch(Exception)
    {
        console.log("Exception is " + Exception);
        console.log("EffictiveService is " + requestUrl);
```

```
        res.writeHead(200, { "Content-Type": "text/html" });
        res.write(requestUrl);
        res.end();
    }
})


function start(route, handle) {

  //  function onRequest(request, response) {

  //  var postData = "";
  //  var pathname = url.parse(request.url).pathname;
  //  console.log("Request for " + pathname + " received.");

  //  request.setEncoding("utf8");

  //  request.addListener("data", function(postDataChunk) {
  //      postData += postDataChunk;

  //    //console.log("Received POST data chunk '"+
  //    //postDataChunk + "'.");
  //  });

  //  request.addListener("end", function () {
  //    route(handle, pathname, response, postData);
  //  });

  //  }

  //http.createServer(onRequest).listen(8888);
    //console.log("Server has started.");

    http.createServer(app).listen(7000);
    console.log("BalanceService has started.");

    Replication();
}

exports.start = start;
```

Модуль: Сервис хранения данных

Файл: server.js

```
var http = require("http");
var url = require("url");

var express = require("express");
```

```
var upload = require("express-fileupload");
var app = express();
app.use(upload());

var fs = require('fs');

var mkdirp = require('mkdirp');

var WebService = "http://localhost";
var WebService_Port = 8888;

var DataService_1 = "http://localhost";
var DataService_1_Port = 7771;

var DataService_2 = "http://localhost";
var DataService_2_Port = 7772;

var currentDate_int = 0;

var DateTimeService = "http://localhost";
var DateTimeService_Port = 7001;

var ExchangeService = "http://localhost";
var ExchangeService_Port = 7002;

var serverIs = true;

function Replication() {
    console.log(' === Replication started === ');

    var folderPath = __dirname + '/uploads/';

    var usersdirs = '';

    fs.readdirSync(folderPath).forEach(file => {
        if (usersdirs != '') {
            usersdirs += '|';
        }

        usersdirs += file;
    })

    if (usersdirs != null && usersdirs != undefined && usersdirs
!= '') {

        usersdirs = usersdirs.split('|');

        usersdirs.forEach(user => {
            console.log('');
            console.log('user: ' + user);

            var userdir = folderPath + user;
```

```
            var allFiles = fs.readFileSync(userdir +
'\\AllFiles.rdb').toString();

            var replicationData = fs.readFileSync(userdir +
'\\Transactions.rpl').toString();
            console.log('replicationData: ' + replicationData);

            var transactions = replicationData.split(';');

            transactions.forEach(function (transaction, i, arr)
{
                if (i != (arr.length - 1)) {
                    var file = transaction.split(':')[0];
                    var action =
transaction.split(':')[1].split('_')[0];
                    var dsidsource =
transaction.split(':')[1].split('_')[1];
                    var dsiddest;

                    if (dsidsource == 1) {
                        dsiddest = 2;
                    }
                    else {
                        dsiddest = 1;
                    }

                    console.log('');
                    console.log('file: ' + file);
                    console.log('action: ' + action);
                    console.log('dsidsource: ' + dsidsource);
                    console.log('dsiddest: ' + dsiddest);

                    console.log('');
                    console.log('make replication');

                    var requesturl;

                    if (dsiddest == 1) {
                        requesturl = DataService_1 + ':' +
DataService_1_Port;
                    }
                    else {
                        requesturl = DataService_2 + ':' +
DataService_2_Port;
                    }

                    if (action == 'add') {
                        requesturl += '/repladd?user=' + user +
'&file=' + file;
                    }
                    else {
```

```
                        requesturl += '/repldel?user=' + user +
'&file=' + file;
                    }

                    console.log('requesturl: ' + requesturl);

                    http.get(requesturl, (resp2) => {

                        let data = '';

                        resp2.on('data', (chunk) => {

                        });

                        resp2.on('end', () => {

                        });

                    }).on("error", (err) => {
                        console.log("Error: " + err.message);
                    });
                }
            });

            transactions.forEach(function (transaction, i, arr)
{
                if (i != (arr.length - 1)) {

                    var file = transaction.split(':')[0];
                    var action =
transaction.split(':')[1].split('_')[0];
                    var dsidsource =
transaction.split(':')[1].split('_')[1];
                    var dsiddest;

                    if (dsidsource == 1) {
                        dsiddest = 2;
                    }
                    else {
                        dsiddest = 1;
                    }

                    if (action == 'add') {
                        allFiles = allFiles.split(file + ':' +
dsidsource + ';');
                        allFiles = allFiles[0] + file + ':12;' +
allFiles[1];
                    }
                    else {

                    }
                }
```

```
            });

            fs.writeFile(userdir + '\\Transactions.rpl', '',
function (err) {
                if (err) throw err;
            });

            console.log("AllFiles: " + allFiles);

            fs.writeFile(userdir + '\\AllFiles.rdb', allFiles,
function (err) {
                if (err) throw err;
            });

        })
    }

    console.log(' === Replication END === ');
    setTimeout(Replication, 20000);
}

function GetUserNameByID(_userID) {
    var accountTable = __dirname + '/DataBase/Accounts.txt';

    if ((fs.existsSync(accountTable))) {
        var userData = fs.readFileSync(accountTable).toString();
        userData = userData.split("\r\n");

        for (var i = 1; i < userData.length; i++) {
            var userID = userData[i].split(' ')[2];

            if (_userID == userID) {
                return userData[i].split(' ')[0];
            }
        }

        return null;

    }
    else {
        return 'error';
    }
}

function GetUserRaitingByID(_userID) {
    var userTable = __dirname + '/DataBase/Users.txt';

    if ((fs.existsSync(userTable))) {
        var userData = fs.readFileSync(userTable).toString();
        userData = userData.split("\r\n");

        for (var i = 1; i < userData.length; i++) {
```

```
            var userID = userData[i].split(' ')[0];

            if (_userID == userID) {
                return userData[i].split(' ')[2];
            }
        }

        return null;

    }
    else {
        return 'error';
    }
}

function GetNextAdviceID() {
    var path = __dirname + '/DataBase/AdviceID.txt';
    var nextAdviceID = fs.readFileSync(path).toString();

    nextAdviceID = parseInt(nextAdviceID);

    var nextAdviceID_new = nextAdviceID + 1;

    fs.writeFileSync(path, nextAdviceID_new);

    return nextAdviceID;
}

function GetItemNameByID(_itemID) {
    var itemsTable = __dirname + '/DataBase/Items.txt';

    if ((fs.existsSync(itemsTable))) {
        var itemData = fs.readFileSync(itemsTable).toString();
        itemData = itemData.split("\r\n");

        for (var i = 1; i < itemData.length; i++) {
            var itemID = itemData[i].split(' ')[0];

            if (_itemID == itemID) {
                return itemData[i].split(' ')[1];
            }
        }

        return null;

    }
    else {
        return 'error';
    }
}

function GetAdvicePriceByID(_adviceID) {
```

```
    var advicesTable = __dirname + '/DataBase/Advices.txt';

    if ((fs.existsSync(advicesTable))) {
        var adviceData =
fs.readFileSync(advicesTable).toString();
        adviceData = adviceData.split("\r\n");

        for (var i = 1; i < adviceData.length; i++) {
            var adviceID = adviceData[i].split(' ')[0];

            if (_adviceID == adviceID) {
                return parseInt(adviceData[i].split(' ')[10]);
            }
        }

        return null;

    }
    else {
        return 'error';
    }
}

app.get('/verifyUser', function (req, res) {

    console.log("verifyUser was called");
    console.log("Params: " + url.parse(req.url).search);

    var login =
url.parse(req.url).search.split('&')[0].split('=')[1];
    var pass =
url.parse(req.url).search.split('&')[1].split('=')[1];

    console.log("login: " + login);
    console.log("pass: " + pass);

    var accountTable = __dirname + '/DataBase/Accounts.txt';

    if ((fs.existsSync(accountTable))) {
        var userData = fs.readFileSync(accountTable).toString();
        userData = userData.split("\r\n");

        for (var i = 1; i < userData.length; i++) {
            console.log("userData: " + userData[i]);

            var checkedLogin = userData[i].split(' ')[0];
            var checkedPass = userData[i].split(' ')[1];
            var userID = userData[i].split(' ')[2];

            console.log("checkedLogin: " + checkedLogin);
            console.log("checkedPass: " + checkedPass);
```

```
            if (login == checkedLogin) {
                console.log("valid user: true");

                if (pass == checkedPass) {
                    console.log("valid pass: true");

                    res.writeHead(200, { "Content-Type":
"text/html" });
                    res.write(userID);
                    res.end();
                    return;
                }
                else {
                    console.log("valid pass: false");

                    res.writeHead(200, { "Content-Type":
"text/html" });
                    res.write('null');
                    res.end();
                    return;
                }
            }
        }

        console.log("valid user: false");

        res.writeHead(200, { "Content-Type": "text/html" });
        res.write('null');
        res.end();
        return;
    }
    else {
        console.log("Database Error: false");
        res.writeHead(200, { "Content-Type": "text/html" });
        res.write('error');
        res.end();
        return;
    }
})

app.get('/getUserInfo', function (req, res) {

    console.log("77777777777777opop getUserInfo!!!!_" +
req.url);

    var userID = url.parse(req.url).search.split("userID=")[1];
    var usersTable = __dirname + '/DataBase/Users.txt';

    if ((fs.existsSync(usersTable))) {
        var userData = fs.readFileSync(usersTable).toString();
        userData = userData.split("\r\n");
```

```
        for (var i = 1; i < userData.length; i++) {
            console.log("userData: " + userData[i]);

            var checkedUserID = userData[i].split(' ')[0];
            console.log("checkedUserID: " + checkedUserID);

            if (userID == checkedUserID) {
                console.log("userID == checkedUserID");

                res.writeHead(200, { "Content-Type": "text/html"
});
                res.write(userData[i]);
                res.end();
                return;
            }
        }

        res.writeHead(200, { "Content-Type": "text/html" });
        res.write('');
        res.end();
        return;
    }
    else {
        console.log("Database Error: false");
        res.writeHead(200, { "Content-Type": "text/html" });
        res.write('error');
        res.end();
        return;
    }


    var allData = fs.readFileSync(userInfo).toString();

    res.writeHead(200, { "Content-Type": "text/html" });
    res.write(allData);
    res.end();
})

app.get('/getAdvices', function (req, res) {

    console.log("opop getAdvices!!!!");

    var allData = fs.readFileSync(__dirname +
'/DataBase/Advices.txt').toString();

    if (allData != null && allData != undefined && allData !=
"") {
        var allDataParts = allData.split("\r\n");

        allData = "";

        for (var i = 1; i < allDataParts.length; i++) {
```

```
            if (i > 1) {
                allData += "\r\n";
            }

            tempData = allDataParts[i].split(" ");

            //Имя и рейтинг эксперта
            var expertID = tempData[1];

            var expertName = GetUserNameByID(expertID);
            if (expertName == null || expertName == 'error') {
                expertName = '<Error>';
            }

            var expertRaiting = GetUserRaitingByID(expertID);
            if (expertRaiting == null || expertRaiting ==
'error') {
                expertRaiting = '<Error>';
            }

            //Продолжительность сессии
            var startDate = tempData[3].split("-").join(".");
            var startDateInt = new Date(startDate.split('.')[0],
startDate.split('.')[1], startDate.split('.')[2], 0, 0, 0, 0);

            var endDate = tempData[4].split("-").join(".");
            var endDateInt = new Date(endDate.split('.')[0],
endDate.split('.')[1], endDate.split('.')[2], 0, 0, 0, 0);

            var sessionLength = parseInt((endDateInt -
startDateInt) / (1000 * 60 * 60 * 24));

            //Advice
            var adviceID = tempData[0];
            var pProfit = tempData[8];
            var pLoss = tempData[9];
            var advicePrice = tempData[10];

            //Формирование данных для ответа
            allData += adviceID + " " + expertRaiting + " " +
expertName + " " + startDate + " " + endDate + " " +
sessionLength + " " + pProfit + " " + pLoss + " " + advicePrice;
        }
    }
    else {
        allData = "";
    }

    console.log('allData: ' + allData);

    res.writeHead(200, { "Content-Type": "text/html" });
    res.write(allData);
```

```
    res.end();
})

app.get('/getBoughtAdvices', function (req, res) {

    console.log("opop getBoughtAdvices2!!!!");

    var usersTable = __dirname + '/DataBase/Users.txt';

    var userID = url.parse(req.url).search.split("userID=")[1];

    if ((fs.existsSync(usersTable))) {
        var userData = fs.readFileSync(usersTable).toString();
        userData = userData.split("\r\n");

        for (var i = 1; i < userData.length; i++) {
            console.log("userData: " + userData[i]);

            var checkedUserID = userData[i].split(' ')[0];
            console.log("checkedUserID: " + checkedUserID);

            if (userID == checkedUserID) {
                console.log("userID == checkedUserID");

                var boughtAdvicesList = userData[i].split('
')[3];

                if (boughtAdvicesList != "-") {
                    boughtAdvicesList =
boughtAdvicesList.split(',');

                    console.log("boughtAdvicesList: " +
boughtAdvicesList);

                    var allData = fs.readFileSync(__dirname +
'/DataBase/Advices.txt').toString();

                    if (allData != null && allData != undefined
&& allData != "") {
                        console.log("allData in normal.");

                        var allDataParts =
allData.split("\r\n");

                        console.log("allDataParts: " + allData);

                        allData = "";

                        for (var j = 1; j < allDataParts.length;
j++) {
                            tempData = allDataParts[j].split("
");
```

```
                            var adviceID = tempData[0];

                            for (var k = 0; k <
boughtAdvicesList.length; k++) {
                                //если этот совет есть среди
списка купленных
                                if (adviceID ==
boughtAdvicesList[k]) {
                                    //TODO: ??? j или k
                                    if (allData != "") {
                                        allData += "\r\n";
                                    }


                                    //Имя и рейтинг эксперта
                                    var expertID = tempData[1];

                                    var expertName =
GetUserNameByID(expertID);
                                    if (expertName == null ||
expertName == 'error') {
                                        expertName = '<Error>';
                                    }

                                    var expertRaiting =
GetUserRaitingByID(expertID);
                                    if (expertRaiting == null ||
expertRaiting == 'error') {
                                        expertRaiting =
'<Error>';
                                    }

                                    //Продолжительность сессии
                                    var startDate =
tempData[3].split("-").join(".");
                                    var startDateInt = new
Date(startDate.split('.')[0], startDate.split('.')[1],
startDate.split('.')[2], 0, 0, 0, 0);

                                    var endDate =
tempData[4].split("-").join(".");
                                    var endDateInt = new
Date(endDate.split('.')[0], endDate.split('.')[1],
endDate.split('.')[2], 0, 0, 0, 0);

                                    var sessionLength =
parseInt((endDateInt - startDateInt) / (1000 * 60 * 60 * 24));

                                    //Advice
                                    var pProfit = tempData[8];
                                    var pLoss = tempData[9];
```

```
                                    var advicePrice =
tempData[10];

                                    //Формирование данных для
ответа
                                    allData += adviceID + " " +
expertRaiting + " " + expertName + " " + startDate + " " +
endDate + " " + sessionLength + " " + pProfit + " " + pLoss + "
" + advicePrice;
                                }
                            }
                        }
                    }
                    else {
                        allData = "";
                    }

                    console.log('allData newnew: ' + allData);

                    res.writeHead(200, { "Content-Type":
"text/html" });
                    res.write(allData);
                    res.end();
                    return;
                }
                else {
                    res.writeHead(200, { "Content-Type":
"text/html" });
                    res.write("");
                    res.end();
                    return;
                }
            }
        }

        res.writeHead(200, { "Content-Type": "text/html" });
        res.write('');
        res.end();
        return;
    }
    else {
        console.log("Database Error: false");
        res.writeHead(200, { "Content-Type": "text/html" });
        res.write('error');
        res.end();
        return;
    }
})

app.get('/saveadvice', function (req, res) {

    console.log("Request handler 'saveAdvice' was called.");
```

```
    console.log("");
    console.log("");
    console.log("request: " + req.url);

    var userID = url.parse(req.url).search.split("userid=")[1];

    var adviceData =
url.parse(req.url).search.split("data=")[1];

    fs.appendFileSync(__dirname + "/DataBase/Advices.txt",
"\r\n" + (GetNextAdviceID() + "|" +
decodeURI(adviceData)).split("|").join(" "));

    res.writeHead(200, { "Content-Type": "text/plain" });
    res.end();
})

app.get('/buyadvice', function (req, res) {

    console.log("opop buyadvice!!!!");

    var responseText = '';

    var userName =
url.parse(req.url).search.split("user=")[1].split('&')[0];
    var userID =
url.parse(req.url).search.split("userid=")[1].split('&')[0];
    var adviceID =
url.parse(req.url).search.split("adviceid=")[1];

    var usersTable = __dirname + '/DataBase/Users.txt';

    if ((fs.existsSync(usersTable))) {
        var userData = fs.readFileSync(usersTable).toString();
        userData = userData.split("\r\n");

        for (var i = 1; i < userData.length; i++) {
            console.log("userData: " + userData[i]);

            var userColumnsValues = userData[i].split(' ');

            var checkedUserID = userColumnsValues[0];

            console.log("checkedUserID: " + checkedUserID);

            if (userID == checkedUserID) {
                console.log("userID == checkedUserID");

                var checkedUserMoney =
parseInt(userColumnsValues[1]);
                var advicePrice = GetAdvicePriceByID(adviceID);
```

```
                if (advicePrice <= checkedUserMoney) {
                    checkedUserMoney -= advicePrice;

                    var newboughtAdvicesList = "";
                    var boughtAdvicesList =
userColumnsValues[3];

                    //Если еще нет приобретенных советов -
добавим этот первым.
                    //В противном случае добавим через запятую
                    if (boughtAdvicesList != "-") {
                        newboughtAdvicesList = boughtAdvicesList
+ "," + adviceID;
                    }
                    else {
                        newboughtAdvicesList = adviceID;
                    }

                    userData[i] = userColumnsValues[0] + " " +
checkedUserMoney + " " + userColumnsValues[2] + " " +
newboughtAdvicesList + " " + userColumnsValues[4];

                    break;
                }
                else {
                    responseText = 'Not enough money to buy this
advice!';
                }
            }
        }

        var userData_new = "";

        for (var i = 0; i < userData.length; i++) {
            userData_new += userData[i];

            if (i != userData.length - 1) {
                userData_new += "\r\n";
            }
        }

        fs.writeFileSync(usersTable, userData_new);

        res.writeHead(200, { "Content-Type": "text/html" });
        res.write(responseText);
        res.end();
        return;
    }
    else {
        console.log("Database Error: false");
        res.writeHead(200, { "Content-Type": "text/html" });
        res.write('error');
```

```
        res.end();
        return;
    }
})

app.get('/getadvice', function (req, res) {

    console.log("opop getAdvice!!!!");

    var userName =
url.parse(req.url).search.split("user=")[1].split('&')[0];
    var userID =
url.parse(req.url).search.split("userid=")[1].split('&')[0];
    var adviceID =
url.parse(req.url).search.split("adviceid=")[1];

    var allData = fs.readFileSync(__dirname +
'/DataBase/Advices.txt').toString();

    if (allData != null && allData != undefined && allData !=
"") {
        var allDataParts = allData.split("\r\n");

        allData = "";

        for (var i = 1; i < allDataParts.length; i++) {
            tempData = allDataParts[i].split(" ");

            var checkedAdviceID = tempData[0];

            if (adviceID == checkedAdviceID) {
                //Имя и рейтинг эксперта
                var expertID = tempData[1];

                var expertName = GetUserNameByID(expertID);
                if (expertName == null || expertName == 'error')
{
                    expertName = '<Error>';
                }

                var expertRaiting =
GetUserRaitingByID(expertID);
                if (expertRaiting == null || expertRaiting ==
'error') {
                    expertRaiting = '<Error>';
                }

                //Продолжительность сессии
                var startDate = tempData[3].split("-
").join(".");
```

```
                var startDateInt = new
Date(startDate.split('.')[0], startDate.split('.')[1],
startDate.split('.')[2], 0, 0, 0, 0);

                var endDate = tempData[4].split("-").join(".");
                var endDateInt = new Date(endDate.split('.')[0],
endDate.split('.')[1], endDate.split('.')[2], 0, 0, 0, 0);

                var sessionLength = parseInt((endDateInt -
startDateInt) / (1000 * 60 * 60 * 24));

                //Advice
                var itemID = tempData[2];
                var itemName = GetItemNameByID(itemID);
                if (itemName == null || itemName == 'error') {
                    itemName = "<Error>";
                }

                var itemPrice = tempData[5];
                var takeProfit = tempData[6];
                var stopLoss = tempData[7];

                var pProfit = tempData[8];
                var pLoss = tempData[9];
                var advicePrice = tempData[10];

                var adviceSatus = tempData[11];
                var endReason = tempData[12];
                var profit = tempData[13];

                //Формирование данных для ответа
                allData += expertName + " " + expertRaiting + "
" + itemName + " " + startDate + " " + endDate + " " +
sessionLength + " " + itemPrice + " " + takeProfit + " " +
stopLoss + " " + pProfit + " " + pLoss + " " + advicePrice + " "
+ adviceSatus + " " + endReason + " " + profit;

                break;
            }
        }
    }
    else {
        allData = "";
    }

    console.log('allData: ' + allData);

    res.writeHead(200, { "Content-Type": "text/html" });
    res.write(allData);
    res.end();
})
```

```
function start(route, handle) {

    http.createServer(app).listen(7000);
    console.log("BalanceService has started.");

}

exports.start = start;
```

Модуль: Сервис верификации рекомендаций

Файл: server.js

```
var http = require("http");
var url = require("url");

var express = require("express");
var upload = require("express-fileupload");
var app = express();
app.use(upload());

var fs = require('fs');

var mkdirp = require('mkdirp');

var WebService = "http://localhost";
var WebService_Port = 8888;

var DataService_1 = "http://localhost";
var DataService_1_Port = 7771;

var DataService_2 = "http://localhost";
var DataService_2_Port = 7772;

var currentDate_int = 0;
//var currentPrice = 515;

var DateTimeService = "http://localhost";
var DateTimeService_Port = 7001;

var ExchangeService = "http://localhost";
var ExchangeService_Port = 7002;

var serverIs = true;

function CheckData() {
    console.log("\nCheckData start");

    //Текущая дата
    http.get(DateTimeService + ':' + DateTimeService_Port +
'/getcurrentdate', (resp) => {
        let data = '';

        resp.on('data', (chunk) => {
            data += chunk;
        });

        resp.on('end', () => {
            var currentDate_new = new
Date(parseInt(parseInt(data * 1)));
```

```
            console.log("dateTime: " + parseInt(currentDate_new
* 1));
            console.log("dateTime: " + currentDate_new.getDate()
+ "." + currentDate_new.getMonth() + "." +
currentDate_new.getFullYear());

            currentDate_new = parseInt(currentDate_new * 1);

            console.log("currentDate_new: " + currentDate_new);
            console.log("currentDate_int: " + currentDate_int);

            if (currentDate_new > currentDate_int) {
                console.log("new date is");

                currentDate_int = currentDate_new;

                //Текущий курс
                http.get(ExchangeService + ':' +
ExchangeService_Port + '/getcurrentprice', (resp) => {
                    let data = '';

                    resp.on('data', (chunk) => {
                        data += chunk;
                    });

                    resp.on('end', () => {
                        var currentPrice = parseFloat(data);
                        console.log("currentPrice555: " +
currentPrice);

                        var allAdvices =
fs.readFileSync(__dirname + '/DataBase/Advices.txt').toString();

                        var flag = false;

                        if (allAdvices != null && allAdvices !=
undefined && allAdvices != "") {
                            var allDataParts =
allAdvices.split("\r\n");

                            console.log("AdvicesCount: " +
(allDataParts.length - 1));

                            for (var i = 1; i <
allDataParts.length; i++) {
                                var columnsValues =
allDataParts[i].split(' ');

                                console.log("Advice ID: " +
columnsValues[0]);
```

```
                                var adviceStatus =
parseInt(columnsValues[11]);

                                if (adviceStatus == 1) {
                                    console.log("adviceStatus ==
1");

                                    var adviceStatus_new = "";
                                    var endReason = "";
                                    var profit = "";

                                    var userID =
columnsValues[1];

                                    var endDate =
columnsValues[4];
                                    var startPrice =
parseInt(columnsValues[5]);
                                    var takeProfit =
parseInt(columnsValues[6]);
                                    var stopLoss =
parseInt(columnsValues[7]);

                                    //var currentDateInt =
currentDate; //new Date(currentDate.split('.')[0],
currentDate.split('.')[1], currentDate.split('.')[2], 0, 0, 0,
0);


                                    var currentDate = new
Date(parseInt(currentDate_int));

                                    console.log("day: " +
currentDate.getDate());

                                    currentDate = new
Date(currentDate.getFullYear(), currentDate.getMonth(),
currentDate.getDate(), 0, 0, 0, 0);
                                    var endDateInt = new
Date(endDate.split('-')[0], parseInt(endDate.split('-')[1]) - 1,
endDate.split('-')[2], 0, 0, 0, 0);

                                    console.log("currentDate: "
+ currentDate);
                                    console.log("endDate: " +
endDate);
                                    console.log("endDateInt: " +
endDateInt);


                                    var dateDifference =
parseInt((endDateInt - currentDate) / (1000 * 60 * 60 * 24));
```

```
                                    console.log("dateDifference:
" + dateDifference);

                                    if ((currentPrice >=
takeProfit) || (currentPrice <= stopLoss) || (dateDifference <=
0)) {
                                        console.log("Change
advice ID: " + columnsValues[0]);

                                        flag = true;
                                        adviceStatus_new = 0;

                                        if (currentPrice >=
takeProfit) {

console.log("currentPrice >= takeProfit");
                                            endReason = 1;
                                            profit =
parseInt(columnsValues[8]); //possibleProfit from advice
                                        }
                                        else {
                                            if (currentPrice <=
stopLoss) {

console.log("currentPrice <= stopLoss");
                                                endReason = 2;
                                                profit =
parseInt(columnsValues[9]) * (-1); //possibleLoss from advice
                                            }
                                            else {
                                                if
(dateDifference <= 0) {

console.log("currentDate >= endDateInt");
                                                    endReason =
3;
                                                    profit =
((currentPrice - startPrice) * 100) / startPrice;
                                                }
                                            }
                                        }

                                        allDataParts[i] =
columnsValues[0] + " " + columnsValues[1] + " " +
columnsValues[2] + " " + columnsValues[3] + " " +
columnsValues[4] + " " + columnsValues[5] + " " +
columnsValues[6] + " " + columnsValues[7] + " " +
columnsValues[8] + " " + columnsValues[9] + " " +
columnsValues[10] + " " + adviceStatus_new + " " + endReason + "
" + profit;
```

```
console.log("allDataParts[i] new: " + allDataParts[i]);

                                        //Начисление кармы
                                        var userTable =
__dirname + '/DataBase/Users.txt';
                                        var userData_new = "";
                                        if
((fs.existsSync(userTable))) {
                                            var userData =
fs.readFileSync(userTable).toString();
                                            userData =
userData.split("\r\n");

                                            userData_new +=
userData[0];
                                            userData_new +=
"\r\n";

                                            for (var j = 1; j <
userData.length; j++) {
                                                var userColumns
= userData[j].split(' ');
                                                var
userID_checked = userColumns[0];

                                                if (userID ==
userID_checked) {
                                                    var
currentRaiting = parseFloat(userColumns[2]);

console.log("currentRaiting: " + currentRaiting);

console.log("profit: " + parseFloat(profit / 100));

                                                    var
newRaiting = currentRaiting + parseFloat(profit / 100);


console.log("newRaiting: " + newRaiting);

                                                    userData[j]
= userColumns[0] + " " + userColumns[1] + " " + newRaiting + " "
+ userColumns[3] + " " + userColumns[4];
                                                }

                                                userData_new +=
userData[j];

                                                if (i !=
userData.length - 1) {
```

```
                                                    userData_new
+= "\r\n";
                                                }
                                            }


fs.writeFileSync(userTable, userData_new);
                                        }
                                    }
                                }
                            }

                            if (flag) {
                                console.log("flag is true");

                                var advicesData_new = "";

                                for (var i = 0; i <
allDataParts.length; i++) {
                                    advicesData_new +=
allDataParts[i];

                                    if (i != allDataParts.length
- 1) {
                                        advicesData_new +=
"\r\n";
                                    }
                                }

                                fs.writeFileSync(__dirname +
'/DataBase/Advices.txt', advicesData_new);
                            }
                        }

                        console.log("CheckData end");
                        setTimeout(CheckData, 50000);
                    });

                }).on("error", (err) => {
                    console.log("Error: " + err.message);
                });
            }
            else {
                console.log("newServiceDate <
currentServiceDate");
                console.log("CheckData end");
                setTimeout(CheckData, 50000);
            }
        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
```

```
    });
}

function start(route, handle) {

    http.createServer(app).listen(7000);
    console.log("CheckDataService has started.");

}

exports.start = start;
```

Модуль: Сервис эмуляции биржи

Файл: server.js

```
var http = require("http");
var url = require("url");

var express = require("express");
var upload = require("express-fileupload");
var app = express();
app.use(upload());

var fs = require('fs');

var DateTimeService = "http://localhost";
var DateTimeService_Port = 7001;

var currentServiceDate = 0;
var newServiceDate = 0;
var currentPrice = 0;

function GetCurrentPrice()
{
    console.log("\n\nGetCurrentPrice start");

    //Текущая дата
    http.get(DateTimeService + ':' + DateTimeService_Port +
'/getcurrentdate', (resp) => {
        let data = '';

        resp.on('data', (chunk) => {
            data += chunk;
        });

        resp.on('end', () => {
            console.log("Getting date succeed");
```

```
            var dateTime = new Date(parseInt(data * 1));

            newServiceDate = parseInt(dateTime * 1);

            if (newServiceDate > currentServiceDate)
            {
                console.log("newServiceDate >
currentServiceDate");

                currentServiceDate = newServiceDate;

                var dateTime_new = "";
                if (parseInt(dateTime.getDate()) < 10) {
                    dateTime_new += "0" + dateTime.getDate();
                }
                else {
                    dateTime_new += dateTime.getDate();
                }

                dateTime_new += ".";

                if (parseInt(dateTime.getMonth() + 1) < 10) {
                    dateTime_new += "0" +
parseInt(dateTime.getMonth() + 1);
                }
                else {
                    dateTime_new += parseInt(dateTime.getMonth()
+ 1);
                }

                dateTime_new += ".";
                dateTime_new += dateTime.getFullYear();

                console.log("dateTime_new: " + dateTime_new);

                var allPrices = fs.readFileSync(__dirname +
'/BitCoin.txt').toString();

                var startIndex =
allPrices.indexOf(dateTime_new);

                console.log("startIndex: " + startIndex);

                if (startIndex > -1) {
                    console.log("Current date is present");

                    //есть такой день
                    allPrices = allPrices.substr(startIndex);
                    allPrices = allPrices.split('\r\n')[0];
                    console.log("currentPrice: " +
allPrices.split(' ')[1]);
```

```
                    currentPrice = parseFloat(allPrices.split('
')[1]);
                }
                else
                {
                    console.log("Current date is NOT present");

                    //Получение предыдущей даты
                    var previosDate = new
Date(parseInt(parseInt(dateTime * 1) - 1 * (1000 * 60 * 60 *
24)));

                    var previosDate_new = "";
                    if (parseInt(previosDate.getDate()) < 10) {
                        previosDate_new += "0" +
previosDate.getDate();
                    }
                    else {
                        previosDate_new +=
previosDate.getDate();
                    }

                    previosDate_new += ".";

                    if (parseInt(previosDate.getMonth() + 1) <
10) {
                        previosDate_new += "0" +
parseInt(previosDate.getMonth() + 1);
                    }
                    else {
                        previosDate_new +=
parseInt(previosDate.getMonth() + 1);
                    }

                    previosDate_new += ".";
                    previosDate_new +=
previosDate.getFullYear();

                    console.log("previosDate: " +
previosDate_new);

                    //получение следующей даты
                    var nextDate = new
Date(parseInt(parseInt(dateTime * 1) + 1 * (1000 * 60 * 60 *
24)));

                    var nextDate_new = "";
                    if (parseInt(nextDate.getDate()) < 10) {
                        nextDate_new += "0" +
nextDate.getDate();
                    }
```

```
                    else {
                        nextDate_new += nextDate.getDate();
                    }

                    nextDate_new += ".";

                    if (parseInt(nextDate.getMonth() + 1) < 10)
{
                        nextDate_new += "0" +
parseInt(nextDate.getMonth() + 1);
                    }
                    else {
                        nextDate_new +=
parseInt(nextDate.getMonth() + 1);
                    }

                    nextDate_new += ".";
                    nextDate_new += nextDate.getFullYear();

                    console.log("nextDate: " + nextDate_new);

                    startIndex =
allPrices.indexOf(previosDate_new);

                    console.log("startIndex: " + startIndex);

                    allPrices = allPrices.substr(startIndex);
                    var previosPrice =
allPrices.split('\r\n')[0].split(' ')[1];
                    var nextPrice =
allPrices.split('\r\n')[1].split(' ')[1];

                    previosPrice = previosPrice.split(',')[0] +
"." + previosPrice.split(',')[1];
                    nextPrice = nextPrice.split(',')[0] + "." +
nextPrice.split(',')[1];

                    console.log("previosPrice: " +
previosPrice);
                    console.log("nextPrice: " + nextPrice);

                    var a = parseFloat(previosPrice);
                    var b = parseFloat(nextPrice);
                    console.log("a: " + a);
                    console.log("b: " + b);
                    var caclulatedPrice = (a + b) / 2;
                    console.log("caclulatedPrice1: " +
caclulatedPrice);

                    currentPrice = caclulatedPrice;
                    //нет такого деня
                    //нужно взять два соседних и апроксимировать
```

```
                }
            }
            else
            {
                console.log("newServiceDate <
currentServiceDate");
            }

            console.log("GetCurrentPrice end");
            setTimeout(GetCurrentPrice, 1000);
        });

    }).on("error", (err) => {
        console.log("Error: " + err.message);
    });
}

app.get('/getcurrentprice', function (req, res) {
    console.log("opop getcurrentprice!!!!");
    res.writeHead(200, { "Content-Type": "text/html" });

    var body = currentPrice;
    body += '';

    console.log("currentPrice: " + body);

    res.write(body);
    res.end();
})

function start(route, handle)
{
    http.createServer(app).listen(7002);

    console.log("ExchangeService has started.");

    GetCurrentPrice();
}

exports.start = start;
```

Модуль: Сервис эмуляции времени

Файл: server.js

```
http = require("http");
var url = require("url");
```

```
var express = require("express");
var upload = require("express-fileupload");
var app = express();
app.use(upload());

var fs = require('fs');

var currentServiceDate = "";
var additionDays = 0;

function GetNextDate()
{
    var currentDate = new Date("2018", "00", "16", 0, 0, 0, 0);


    var newDate = parseInt(parseInt(currentDate * 1) +
(additionDays) * (1000 * 60 * 60 * 24));
    newDate = new Date(newDate);

    var newDateInt = parseInt(newDate * 1);
    currentServiceDate = newDateInt;

    currentDate = new Date(currentServiceDate);

    console.log("currentServiceDate: " + currentDate);
    console.log("currentServiceDateInt: " +
parseInt(currentServiceDate * 1));
}

app.get('/getnextdate', function (req, res) {
    console.log("opop getnextdate!!!!");
    res.writeHead(200, { "Content-Type": "text/html" });

    var param =
parseInt(url.parse(req.url).search.split('days=')[1]);
    console.log("param: " + param);
    additionDays += param;
    GetNextDate();
    res.end();
    return;
})

app.get('/getcurrentdate', function (req, res) {
    console.log("opop getcurrentdate!!!!");
    res.writeHead(200, { "Content-Type": "text/html" });

    var body = currentServiceDate;
    body += '';

    console.log(body);
```

```
    res.write(body);
    res.end();
    return;
})

function start(route, handle)
{
    http.createServer(app).listen(7001);
    console.log("DateTimeService has started.");

    GetNextDate();
}

exports.start = start;
```

Приложение Б. Презентация Microsoft Power Point

Рисунок Б.1 – Слайд 1

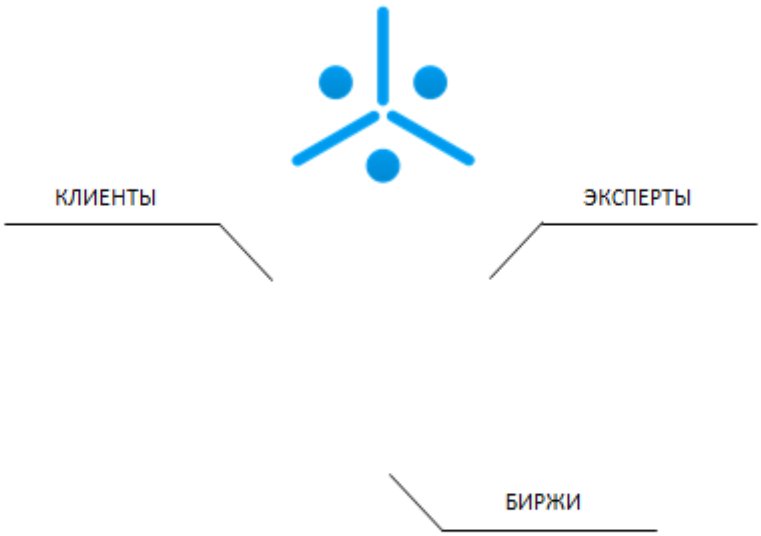

Рисунок Б.2 – Слайд 2

КЛИЕНТЫ

- Физическое лицо;
- Инвестируют;
- Ожидают максимальную прибыль;
- Рискуют личными финансовыми ресурсами.

Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

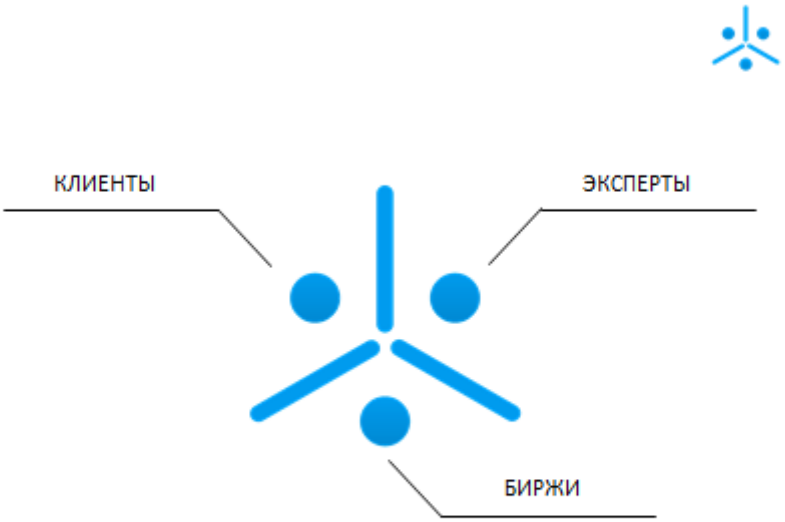

Рисунок Б.6 – Слайд 6

ЭКСПЕРТЫ

- Физическое лицо;
- Имеют цель, направленную на обогащение;
- Ограждены от финансовых рисков;
- Рискуют рейтингом эксперта.

Рисунок Б.7 – Слайд 7

Рисунок Б.8 – Слайд 8

Рисунок Б.9 – Слайд 9

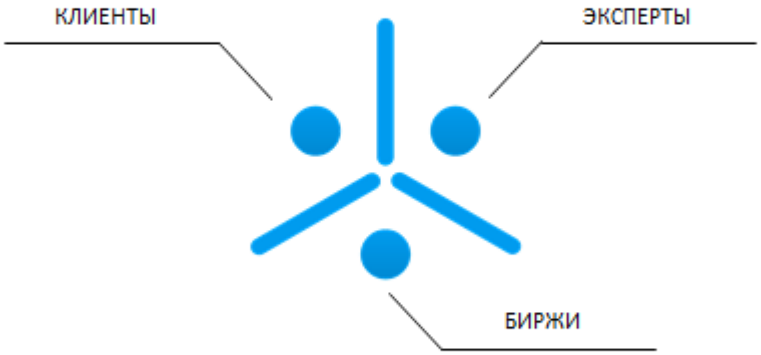

Рисунок Б.10 – Слайд 10

- Юридическое лицо;
- Обеспечивает регулярное функционирование организованного рынка биржевых товаров, валют, ценных бумаг и производных финансовых инструментов;
- Предоставляет API для взаимодействия через Интернет.

БИРЖИ

Рисунок Б.11 – Слайд 11

АРХИТЕКТУРА СИСТЕМЫ

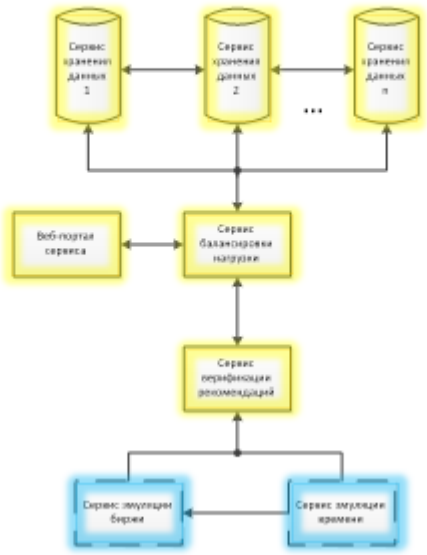

Рисунок Б.12 – Слайд 12

Рисунок Б.13 – Слайд 13

Рисунок Б.14 – Слайд 14

ТЕСТИРОВАНИЕ

Модульное тестирование:

- работа веб-портала;
- работа подсистемы управления хранилищем данных;
- реализация логики маршрутизации и распределения нагрузки;
- эмуляция биржи криптовалюты;
- эмуляция времени системы;

Интеграционное тестирование:

- Тестирование интерфейсов и связей между слоями и компонентами сервиса.

Системное тестирование:

- Проверка соответствия законченного продукта сформированным ранее требованиям.

Рисунок Б.15 – Слайд 15

ТЕСТИРОВАНИЕ

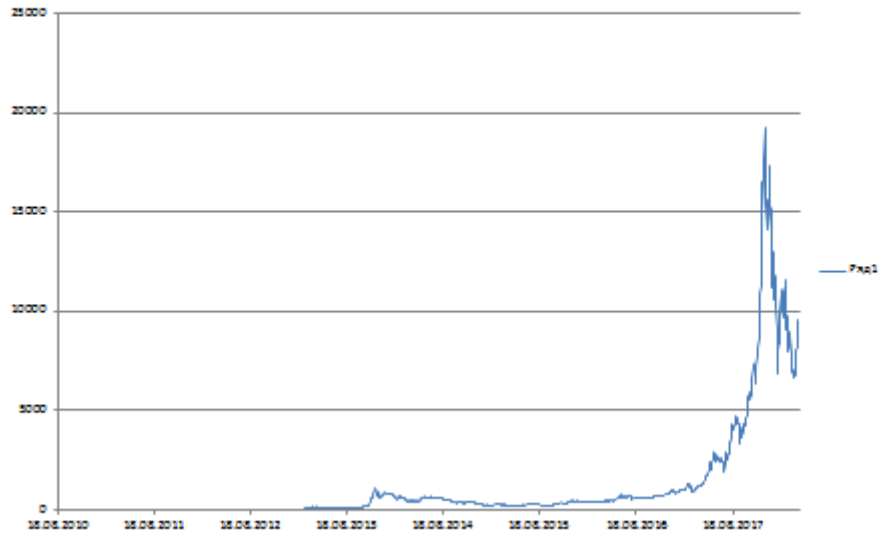

Рисунок Б.16 – Слайд 16

Рисунок Б.17 – Слайд 17

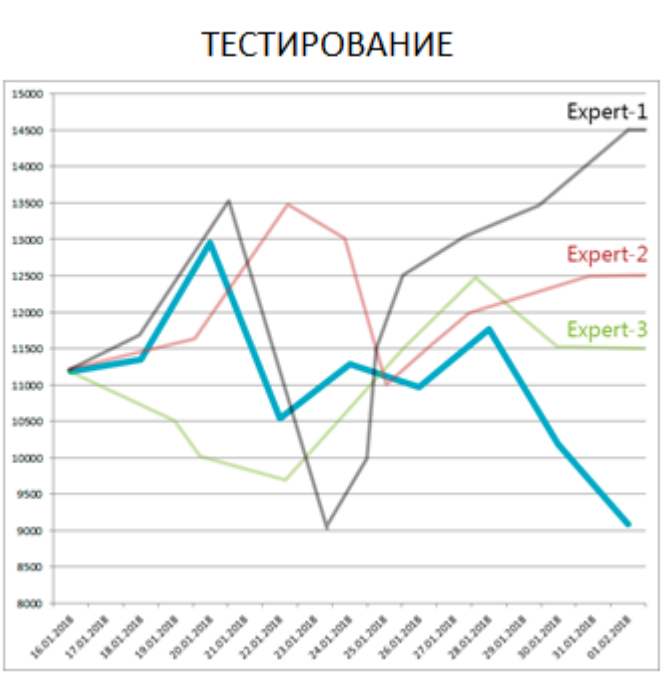

Рисунок Б.18 – Слайд 18

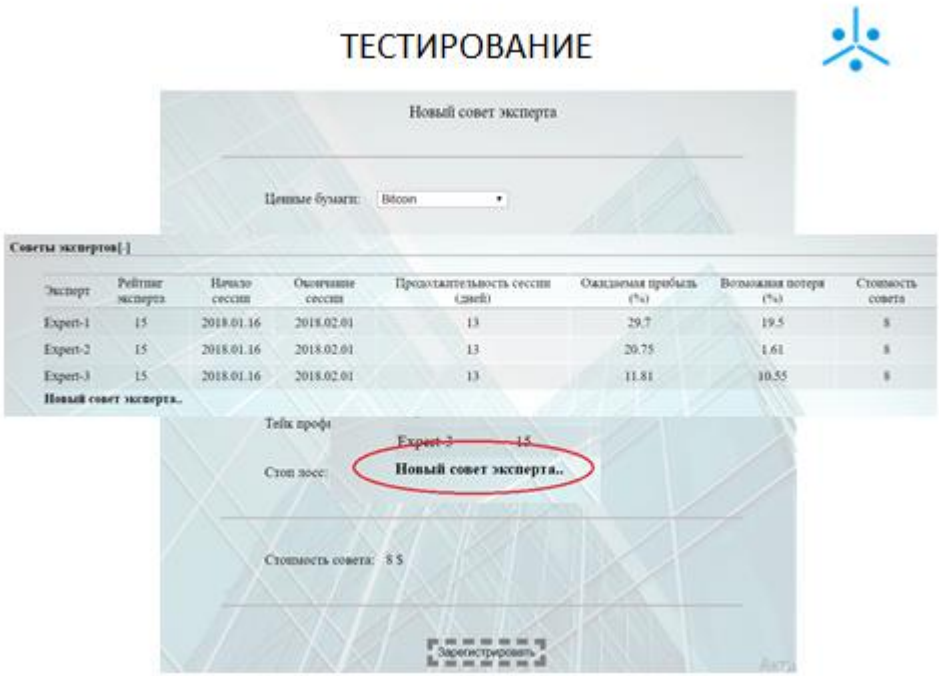

Рисунок Б.19 – Слайд 19

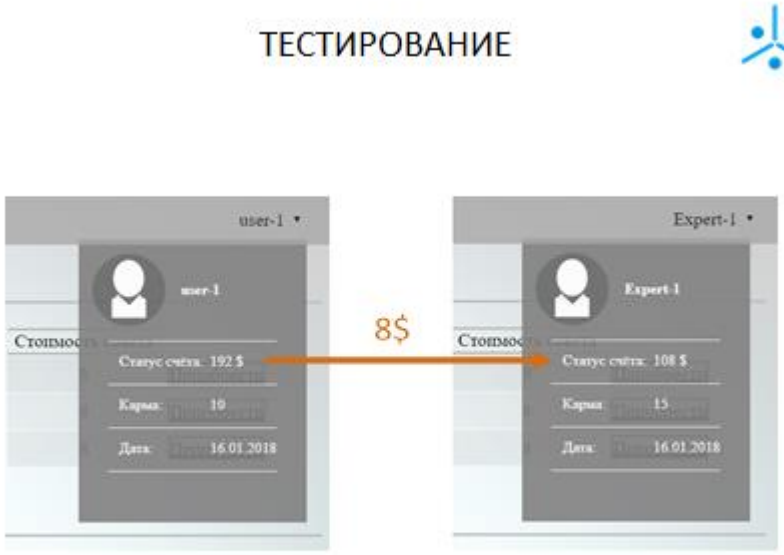

Рисунок Б.20 – Слайд 20

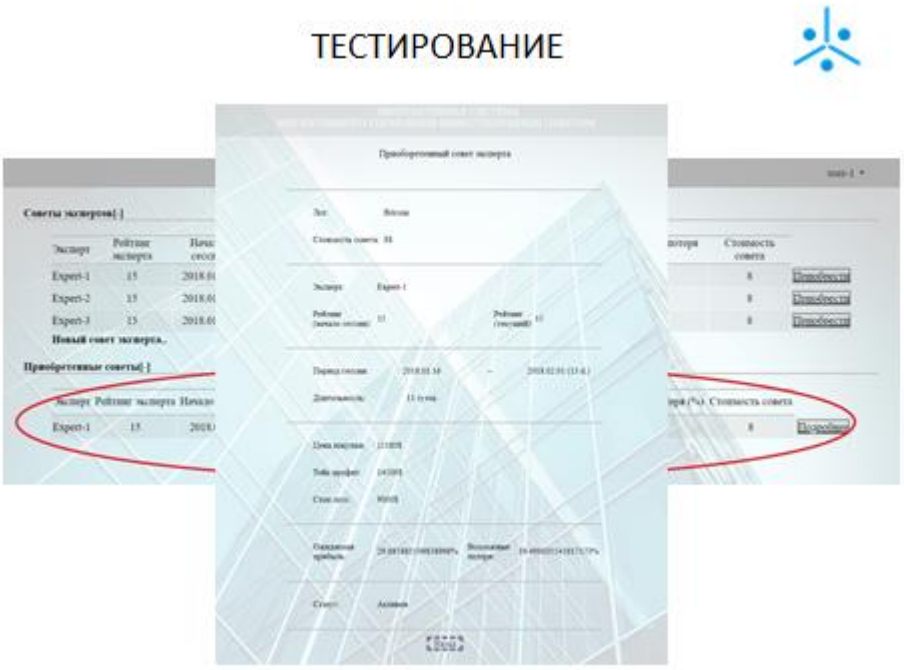

Рисунок Б.21 – Слайд 21

Рисунок Б.22 – Слайд 22

Рисунок Б.23 – Слайд 23

Рисунок Б.24 – Слайд 24

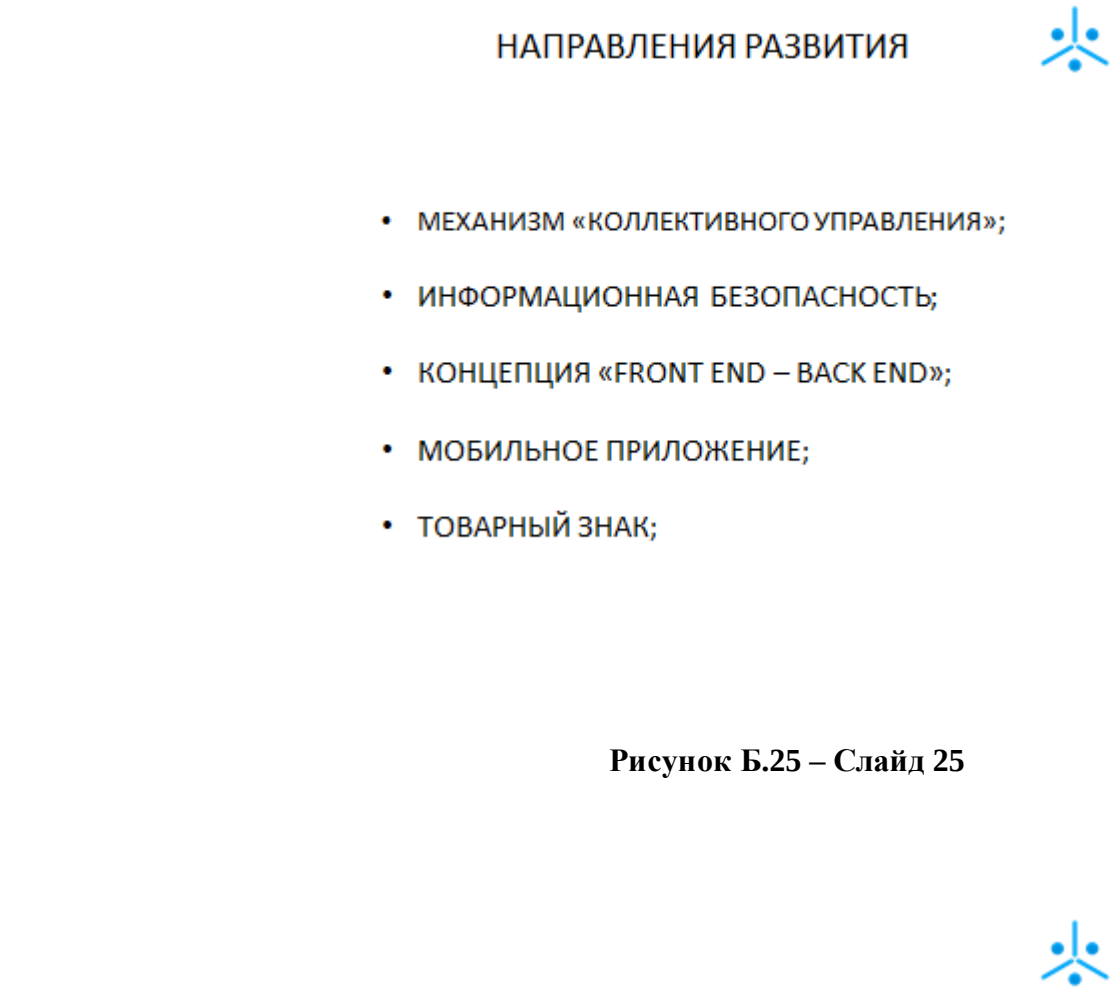

Рисунок Б.25 – Слайд 25

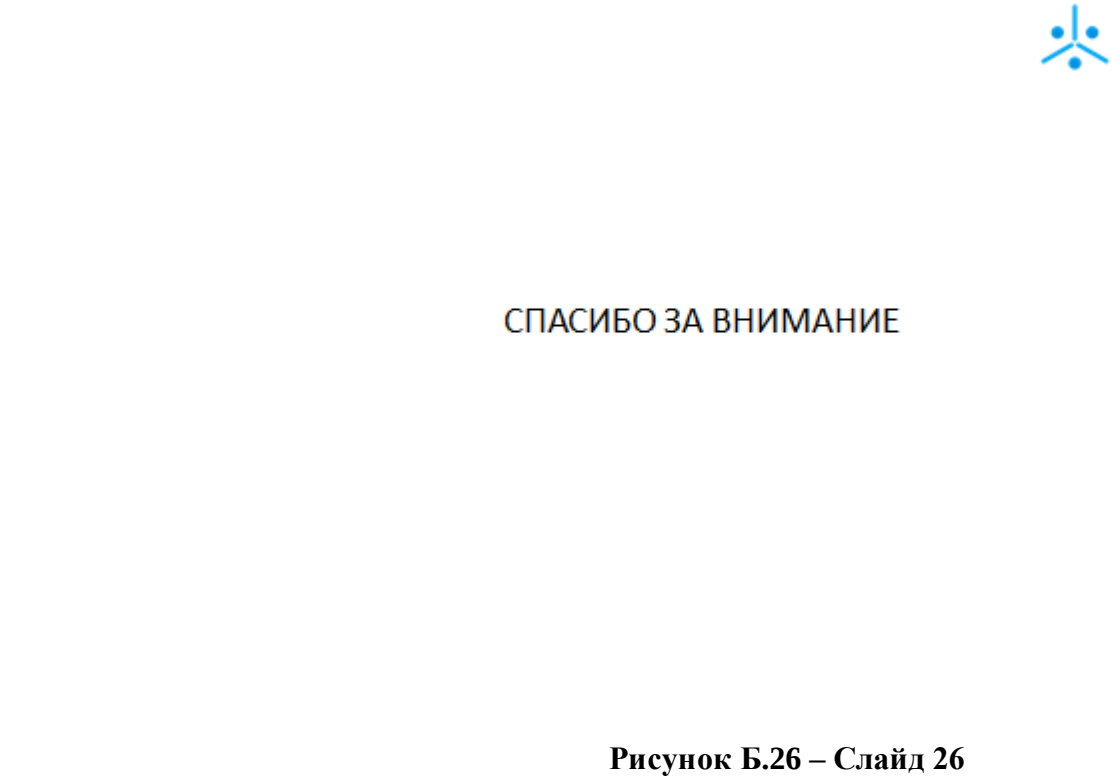

Рисунок Б.26 – Слайд 26

Рисунок Б.27 – Слайд 27