

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математическое обеспечение и администрирование информационных систем

Владимир Андреевич Лямин

Создание платформы для трейдинга «Trading Station»

Отчет по учебной практике

Научный руководитель:
доцент каф. СП, к.т.н. Ю.В. Литвинов

Консультант:
ст. разработчик ООО «Ланит-Терком» С.А. Рябцев

Санкт-Петербург
2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 Постановка задачи.....	4
2 Обзор аналогов.....	5
3 Общая архитектура решения.....	6
4 Описание решения.....	8
4.1 Описание решения деактивации и изменения учетных записей пользователей.....	8
4.2 Интеграционное тестирование.....	9
4.3 Разработка части графических компонентов.....	10
4.4 Тестирование работы валидаторов.....	11
4.5 Получение курса валют из ЦБ.....	11
4.6 Апробация.....	11
ЗАКЛЮЧЕНИЕ.....	12
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	13

ВВЕДЕНИЕ

В настоящее время разработка более или менее крупного проекта проходит через несколько стадий. Например: выяснение требований, планирование и т.д. Поэтому проект «Trading Station» призван ознакомить и научить студентов процессу коммерческой разработки программного обеспечения.

В том числе каждый участник команды должен был попробовать себя в разных сферах. Например все мы были две недели лидерами команды. Лидер должен был распределять новые задачи между членами команды, а также контролировать, чтобы задачи выполнялись. В конце недели он предоставлял отчет консультанту о проделанной работе. Также обязательно надо было посещать совещания по проекту два раза в неделю.

Была выбрана тема трейдинга, так как она позволяет освоить наибольшее количество инструментов. Также нельзя не отметить большой спрос на такие платформы в мире [1], поэтому есть большая вероятность того, что опыт, полученный в процессе разработки, можно будет применить в будущем.

Разрабатываемая система должна быть интуитивно понятной пользователю и легка в обращении. В ней пользователь может приобретать акции любой компании или при помощи бота продавать свои акции на бирже. Приложение позволяет пользователю следить за биржевыми новостями и текущими курсами валют.

1 Постановка задачи

Проект был реализован группой под общим руководством консультанта Рябцева Спартака Александровича, старшего разработчика ООО «Ланит-Терком». Общая цель проекта: создать приложение, которое содержит инструменты для создания и запуска работающих торговых ботов. Для достижения поставленной цели были поставлены следующие общие задачи:

- а) проектирование архитектуры приложения;
- б) создание логики:
 - хранения и управления данными о пользователях;
 - взаимодействия с программным интерфейсом (API) брокера;
 - хранения и управления действиями пользователей с финансовыми инструментами;
 - работы ботов;
 - логирования;
 - валидации данных;
- в) проведение модульного тестирования;
- г) проведение интеграционного тестирования системы;
- д) развертывание платформы на рабочем сервере.

Перед автором стояли следующие частные задачи:

- а) программирование логики деактивации и изменения пользователя;
- б) получение курса валют из ЦБ;
- в) тестирование работы валидаторов;
- г) создание части компонентов графического интерфейса;
- д) проведение интеграционного тестирования;
- е) проведение апробации.

2 Обзор аналогов

При разработке мы провели анализ существующих аналогов и реализовали в рамках проекта наиболее важные детали, которые должны быть в любом приложении для трейдинга. Одним из таких аналогов стал MetaTrader [2] — программное обеспечение, которое относится к классу торговых терминалов и используется трейдерами для доступа к биржам и заключения различных сделок. К достоинствам MetaTrader, по нашему мнению, можно отнести его гибкость и расширяемость, а именно возможность создания собственных сценариев и инструментов. В данном проекте были реализованы таблица курсов валют и график стоимости каждого инструмента из программы MetaTrader (рисунок 1). Так как команда не обладала достаточными знаниями в этой области и временем для разработки приложения такого уровня, то мы решили максимально упростить график цены за акции компании, а также работу ботов, которые имеют всего два правила.

В качестве второго аналога был взят сайт Investing.com [3], который предназначен для оперативного доступа к информации о компаниях и акциях, курсах валют и стоимости сырья. К его недостаткам можно отнести изобилие рекламы. Из знакомства с Investing.com была взята идея, которая заключается в предъявлении пользователю экономических новостей для лучшего понимания рынка и более успешного трейдинга.



Рисунок 1 — Интерфейс приложения «MetaTrader»

3 Общая архитектура решения

Для приложения вместо монолита мы выбрали микросервисную архитектуру, так как она дает ряд следующих преимуществ [4]: масштабируемость и изолируемость. Кроме того, микросервисная архитектура позволяет разделить разработку на несколько команд и обновлять сервисы независимо друг от друга. Связи между сервисами проекта (синие прямоугольники), глобальными сервисами (черные овалы) и графическим интерфейсом пользователя (черный прямоугольник) показаны на рисунке 2.

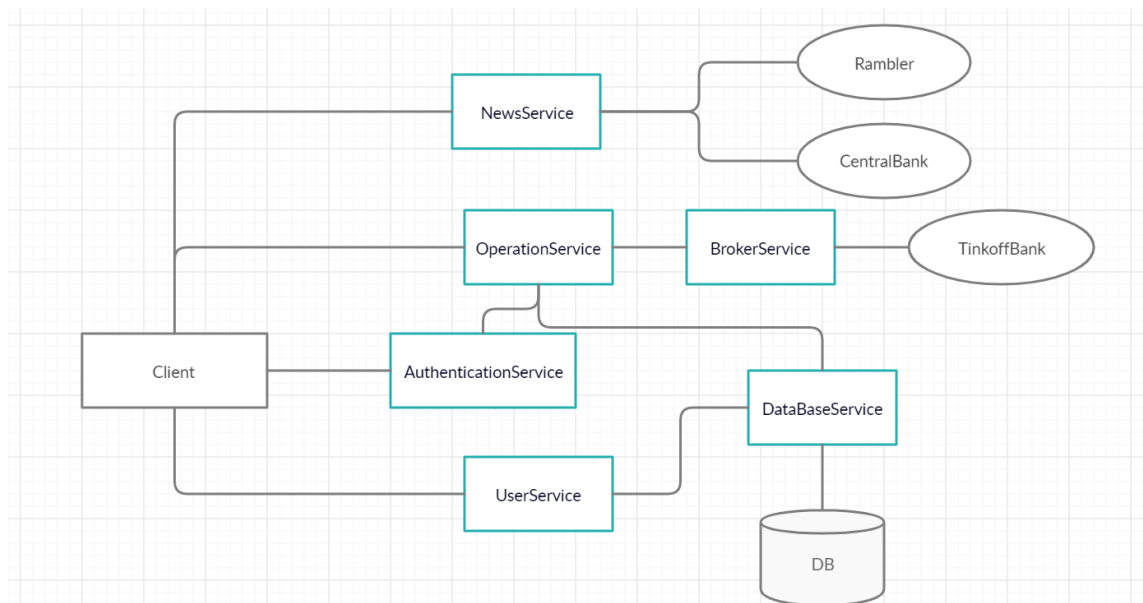


Рисунок 2 — Связи между сервисами приложения «Trading Station»

Сервис «NewsService» использован для получения новостей и курса валют. Его источниками данных выступают системы «Rambler» и «Центробанк». Данные поступают в формате RSS (формат для передачи новостей) [5].

Основная цель сервиса «AuthenticationService» — инкапсулировать логику подтверждения личности пользователя. Сервис производит проверку и выдачу индивидуального токена, а также выполняет вход и выход пользователя из системы.

Сервис «UserService» управляет данными о пользователях системы. Он может зарегистрировать пользователя, изменить или получить информацию о нем. Также он нужен для активации аккаунта и для подтверждения регистрации пользователя при помощи отправки сообщения на почту.

Сервис «DatabaseService» представляет единый интерфейс для работы с базой данных проекта. Его основные функции:

- исполнение CRUD операции с пользователями, брокерами и ботами;
- запись логов;

- миграция.

Сервис «BrokerService» предназначен для взаимодействия с биржевыми брокерами. В его функции входят: создание соединения с нужным брокером, получение списка инструментов, предоставляемого брокером, отслеживание информации об инструментах в реальном времени, реализация торговых сделок.

Сервис «OperationService» предназначен для управления финансовыми операциями пользователей. В рамках этого сервиса осуществляется контроль за операциями покупок и продаж инструментов, контроль баланса пользователя, взаимодействие с портфелем пользователя и взаимодействие с торговыми ботами.

4 Описание решения

4.1 Описание решения деактивации и изменения учетных записей пользователей

Работа над этой задачей осуществлялась в сервисе «UserService». Для единообразия кода было решено использовать специальную структуру запросов. Например, запрос «EditUserRequest» состоит из подзапросов «UserInfoRequest», «PasswordChangeRequest», «AvatarChangeRequest». Более подробно можно увидеть на рисунке 3.

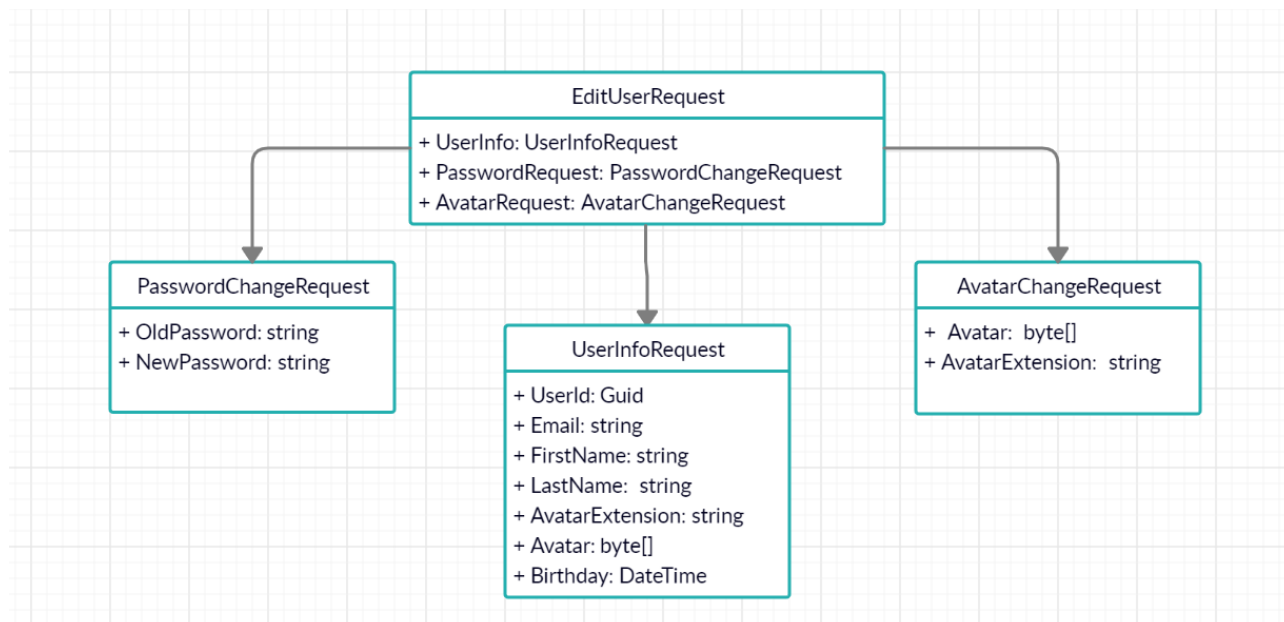


Рисунок 3 — Диаграмма классов запроса к UserService по изменению пользователя

Для решения задачи изменения информации о пользователе был создан класс «EditUserCommand», наследуемый от интерфейса «IEditUserCommand». При инициализации класса в его конструктор передаются различные запросы, например ILogger, который передает результат об удавшемся или неудавшемся изменении данных пользователя.

Когда пользователь хочет изменить информацию о себе, вызывается метод «execute». Единственный параметр — это модель изменения данных пользователя. Метод проверяет, является ли старый пароль пустым. Если да, то он понимает, что изменять его не надо и передает в качестве параметра (UserPasswords) запроса к сервису, отвечающему за базу данных, пустое множество, иначе передается указанный пользователем пароль. Параметры «User» и «UserAvatar» будут заполнены в любом случае. После этого вызывается метод «EditUser», который передает управление сервису «DataBaseService». Он возвращает булево значение: получилось ли изменить данные пользователя или нет.

Тестирование классов «EditUserCommand», «DeleteUserCommand» модульными тестами представляет некоторые проблемы, так как методы ждут ответа от базы данных. Для этого понадобилось использовать mock-объекты [6]. Это объект «заглушка» и ее задачей является имитация действий другого сервиса.

Перед началом тестирования в отдельном методе инициализировались mock-объекты для разных запросов, таких как «UserInfoRequest», «PasswordChangeRequest» и др. Далее нужно выполнить команду setup, которая говорит, что возвращает объект в зависимости от того, что он принимает. В нашем случае метод должен получать «true», если все прошло хорошо, или «false», если произошло исключение. В этом же методе инициализируется модель «UserInfoRequest», так как корректную работу этого запроса проверяют другие тесты, а значит можно считать, что он работает правильно и не тестировать случаи с неправильным заполнением полей. После выполнения этих действий остается только написать тесты обычным образом.

4.2 Интеграционное тестирование

В задачу входило интеграционное тестирование backend-части нашего приложения. Для этого использовали инструмент-тестировщик Postman [7]. Этот инструмент предназначен для тестирования сервисов. В Postman создавались http-запросы, с которыми клиент может обратиться к сервису через REST API.

Приведем пример создания теста на удаление пользователя в Postman. Для начала нужно создать коллекцию. В данном случае она называлась «UserService». Дальше создавались запросы для тестирования разных методов. Выбор типа запроса осуществлялся на основе информации из контроллера сервиса. Например, для деактивации учетной записи пользователя тип HTTP-запроса будет иметь значение «DELETE», а рядом надо указать адрес, на который будет посылаться запрос. На вкладке «Params» указывается идентификатор пользователя, учетную запись которого хотим удалить, а на вкладке «Headers» необходимо указать токен пользователя. В данном случае вкладку «Body» оставляем пустой, в отличие от теста на изменение данных пользователя, когда здесь помещается новая информация о нем. На вкладке «Tests» надо написать, что должно прийти от сервиса в качестве ответа (код результата выполнения запроса). На рисунке 4 показан пример работы теста в Postman.

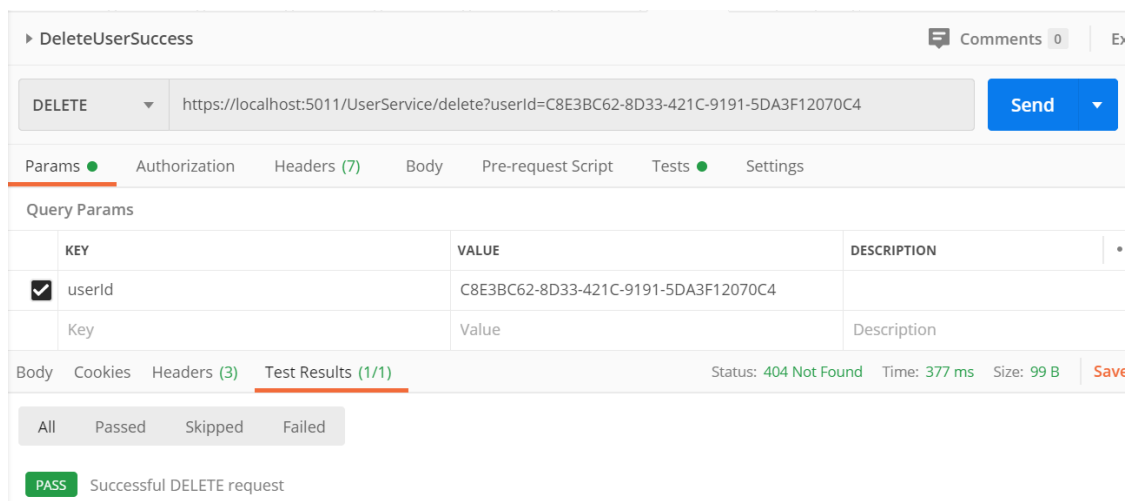


Рисунок 4 — Успешный Postman тест

4.3 Разработка части графических компонентов

В процессе создания приложения понадобилось разработать дизайн сайта. Требовалось сделать основные компоненты главного окна: «Header», в котором будет аватар пользователя, эмблема «Trading Station», кнопки для входа и выхода из приложения, а также перехода на другие страницы (рисунок 5). Также нужно было сделать всплывающее окно для входа пользователя (рисунок 6). В проекте использовался фреймворк «Blazor» [8] для создания интерактивных приложений, которые могут работать на платформе .NET как на стороне сервера, так и на стороне клиента. Все элементы заголовка прописаны в файле «Header.razor». Так как требовались кнопки для разных частей графического интерфейса, то они были разбиты на группы. Например, группа кнопок «header-right» включала кнопки: «Sign Up», «Sign In», «Sign Out», «User Info». Настройка модального окна для входа пользователя в систему делается в файле «SignUpWizzard.razor». Единственным отличием описания модального окна от разработки заголовка или размещения окна валют является наличие команды «modal.Show<>».

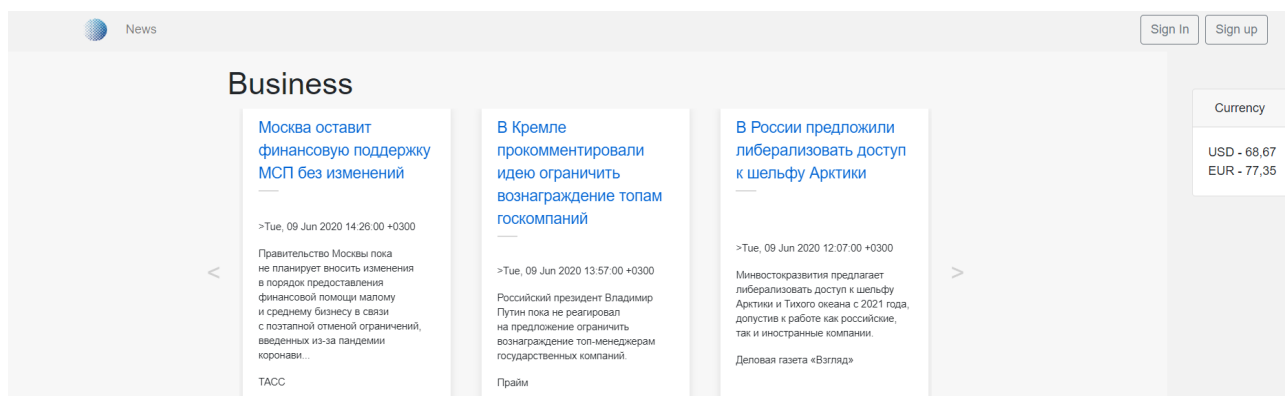


Рисунок 5 — Главная страница приложения

This is a screenshot of a 'Sign In' modal window. It has a title bar with 'Sign In' and a close button (x). Below the title, there are two input fields: 'Email' with the value 'LyaminVA@yandex.ru' and 'Password' with masked characters '....'. At the bottom, there is a blue button labeled 'Sign in'.

Рисунок 6 — Модальное окно для входа пользователя

4.4 Тестирование работы валидаторов

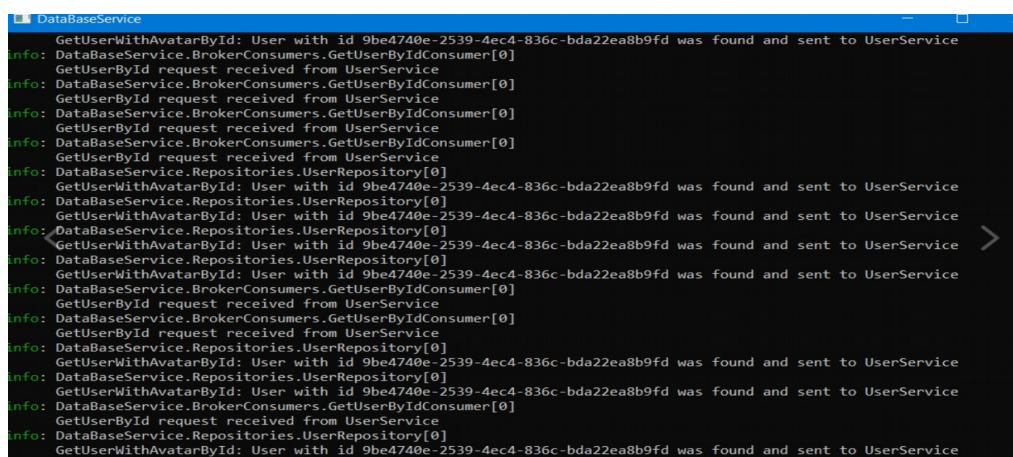
При разработке приложения использовались валидаторы [9]. Валидатор — компьютерная программа, которая проверяет соответствие какого-либо документа, потока данных или фрагмента кода определённому формату, проверяет синтаксическую корректность документа или файла. Были протестированы следующие валидаторы: «PasswordChangeRequestValidator», «UserInfoRequestValidatorTests». В первом тесте проверена работоспособность приложения в следующих ситуациях: все поля пустые, старый паспорт пустой, новый паспорт пустой, оба паспорта одинаковы, внесены корректные данные. Во втором тесте затронуты следующие ситуации: все поля пустые, потерял идентификатор и почтовый адрес пользователя, фамилия и имя слишком длинные или содержат цифры, возраст меньше 18 лет, внесены корректные данные.

4.5 Получение курса валют из ЦБ

В задачу входило получение курса валют из ЦБ. Основной результат решения этой задачи — класс «RussianCBInfo». Для начала надо узнать URI ЦБ. По нему сервис будет получать данные. При инициализации класса «RussianCBInfo» происходит вызов метода «GetData», который отправляет с помощью URI запрос к ЦБ и получает значения курсов наиболее популярных валют: доллар США и евро. Затем при вызове метода «GetCurrencies» данные записываются в список. В каждом узле этого списка есть код значения и само значение.

4.6 Апробация

Разработанный сервис прошел апробацию, все запросы на внесение изменений в код проекта были одобрены четырьмя членами команды. Также проект прошел защиту, во время которой была продемонстрирована его работоспособность. В тестировании сервиса во время защиты приняло участие более десяти экспертов. На рисунке 7 показаны запросы к базе данных приложения на удаленном сервере при защите проекта в ООО «Ланит-Терком».



```
DataBaseService
GetUserWithAvatarById: User with id 9be4740e-2539-4ec4-836c-bda22ea8b9fd was found and sent to UserService
Info: DataBaseService.BrokerConsumers.GetUserByIdConsumer[0]
GetUserById request received from UserService
Info: DataBaseService.BrokerConsumers.GetUserByIdConsumer[0]
GetUserById request received from UserService
Info: DataBaseService.BrokerConsumers.GetUserByIdConsumer[0]
GetUserById request received from UserService
Info: DataBaseService.BrokerConsumers.GetUserByIdConsumer[0]
GetUserById request received from UserService
Info: DataBaseService.Repositories.UserRepository[0]
GetUserWithAvatarById: User with id 9be4740e-2539-4ec4-836c-bda22ea8b9fd was found and sent to UserService
Info: DataBaseService.Repositories.UserRepository[0]
GetUserWithAvatarById: User with id 9be4740e-2539-4ec4-836c-bda22ea8b9fd was found and sent to UserService
Info: DataBaseService.Repositories.UserRepository[0]
GetUserWithAvatarById: User with id 9be4740e-2539-4ec4-836c-bda22ea8b9fd was found and sent to UserService
Info: DataBaseService.Repositories.UserRepository[0]
GetUserWithAvatarById: User with id 9be4740e-2539-4ec4-836c-bda22ea8b9fd was found and sent to UserService
Info: DataBaseService.BrokerConsumers.GetUserByIdConsumer[0]
GetUserById request received from UserService
Info: DataBaseService.BrokerConsumers.GetUserByIdConsumer[0]
GetUserById request received from UserService
Info: DataBaseService.Repositories.UserRepository[0]
GetUserWithAvatarById: User with id 9be4740e-2539-4ec4-836c-bda22ea8b9fd was found and sent to UserService
Info: DataBaseService.BrokerConsumers.GetUserByIdConsumer[0]
GetUserById request received from UserService
Info: DataBaseService.Repositories.UserRepository[0]
GetUserWithAvatarById: User with id 9be4740e-2539-4ec4-836c-bda22ea8b9fd was found and sent to UserService
```

Рисунок 7 — Запросы к сервису базы данных

ЗАКЛЮЧЕНИЕ

В ходе работы над данным проектом были достигнуты следующие результаты:

- реализована логика деактивации и изменения пользователя;
- реализована функция получения курса валют из ЦБ;
- произведено модульное тестирование работы валидаторов;
- реализована часть графических компонентов;
- проведено интеграционное тестирование;
- проведена апробация.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Посещаемость сайтов онлайн-трейдинга резко выросла из-за кризиса, URL:
<https://habr.com/ru/post/294872> (дата обращения 2020-06-08)
2. Букунов С.В., Климин П.Ю. Автоматизированная торговая система для работы на финансовых рынках // Инженерный вестник Дона. — 2019. — № 4 (55). — С. 32-47.
3. Обзор сайта ru.investing.com, URL: http://cdelet.ru/novostnaya_lenta_1/obzor-sayta-ruinvestingcom/ (дата обращения 2020-06-12)
4. Создание микросервисов / С. Ньюмен. — СПб.: Питер, 2016. — 304 с
5. Блоги и RSS: интернет-технологии нового поколения / В.А. Герасевич. - СПб.: БХВ-Петербург, 2006. - 256 с.
6. Экстремальное программирование. Разработка через тестирование / К. Бек. - СПб.: Питер, 2017. - 291 с.
7. Как тестировать API, или Postman для чайников, URL:
<https://geekbrains.ru/posts/kak-testirovat-api-ili-postman-dlya-chajnikov>
(дата обращения 2020-06-09)
8. Введение в ASP.NET Core Blazor, URL:
<https://docs.microsoft.com/ru-ru/aspnet/core/blazor/?view=aspnetcore-3.1>
(дата обращения 2020-06-09)
9. V for Validator, URL:
<https://habr.com/ru/post/348530> (дата обращения 2020-06-09)