Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (ТПУ)

Институт кибернетики

Направление 09.04.01 «Информатика и вычислительная техника»

Кафедра автоматики и компьютерных систем (АИКС)

ПО ДИСЦИПЛИНЕ «ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

Пояснительная записка

Студент гр. 8ВМ71 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю. Ю. Ибетуллов

Студент гр. 8ВМ71 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. И. Пономарев

Студент гр. 8ВМ71 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. А. Рачкован

Студент гр. 8ВМ71 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. А. Черных

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2017 г.

Руководитель:

Инж. кафедры ПИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. Н. Лайком

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2017 г.

**Оглавление**

[Введение 3](#_Toc502299734)

[1 Требования к программе 4](#_Toc502299735)

[1.1 Назначение программы 4](#_Toc502299736)

[1.2 Область применения 4](#_Toc502299737)

[1.3 Варианты использования 4](#_Toc502299738)

[2 Анализ 12](#_Toc502299739)

[2.1 Классы анализа 12](#_Toc502299740)

[2.2 Диаграмма последовательностей классов анализа 14](#_Toc502299741)

[2.3 Диаграммы деятельности 14](#_Toc502299742)

[2.4 Конечные автоматы для динамических классов анализа 14](#_Toc502299743)

[2.5 Пакеты системы 15](#_Toc502299744)

[3 Проектирование 16](#_Toc502299745)

[3.1 Проектные классы 16](#_Toc502299746)

[3.2 Диаграммы последовательностей для операций проектных классов 18](#_Toc502299747)

[4 Реализация 21](#_Toc502299748)

[4.1 Интеграционное тестирование системы 21](#_Toc502299749)

[4.2 Непрерывная интеграция 22](#_Toc502299750)

[4.2.1 Travis CI 22](#_Toc502299751)

[4.2.2 AppVeyor 24](#_Toc502299752)

[5 Документация 28](#_Toc502299753)

[Заключение 29](#_Toc502299754)

[Список использованных источников 30](#_Toc502299755)

# Введение

# Требования к программе

## Назначение программы

## Область применения

## Варианты использования

Варианты использования, или прецеденты – это способ записи требований (Рисунок 1.1).

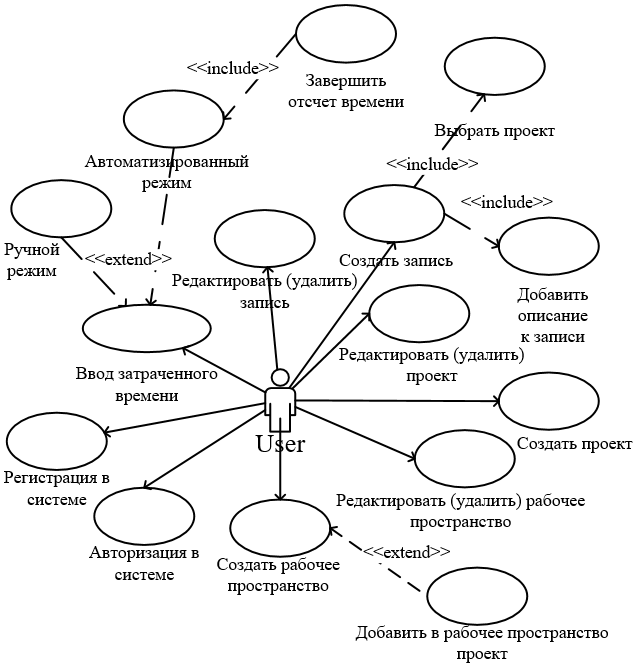


Рисунок 1.1 – Диаграмма вариантов использования

ОПИСАТЬ???

Таблица 1.1 - Варианты использования

|  |  |
| --- | --- |
| Регистрация в системе |  |
| Авторизация в системе |  |
| Создание рабочего пространства |  |
| Редактирование (удаление) рабочего пространства |  |
| Создание проекта |  |
| Редактирование (удаление) проекта |  |
| Создание записи |  |
| Редактирование (удаление) записи |  |
| Ввод рабочего времени |  |

Таблица 1.2 - Прецедент: Регистрация в системе

|  |
| --- |
| ID: 1 |
| *Краткое описание:*  Пользователь проходит регистрацию, вводит свои данные, чтобы в дальнейшем он мог принимать участие в проектах. |
| *Актёры:*  Регистрацию должны пройти все актеры. |
| *Предусловия:*  Пользователь не зарегистрирован. |
| *Основной поток:*   1. Прецедент начинается, когда пользователь переходит на форму регистрации. 2. Пользователь вводит личные данные. 3. Введенные данные проверяются на корректность. 4. Если введенные данные корректны, то регистрация завершается, иначе пользователю указываются ошибки, которые необходимо исправить. |
| *Постусловия:*  Пользователь может быть авторизован. |
| *Альтернативные потоки:*  Нет. |

Таблица 1.3 - Прецедент: Авторизация в системе

|  |
| --- |
| ID: 2 |
| *Краткое описание:*  Пользователь вводит свои данные, чтобы войти в систему и использовать функционал приложения. |
| *Актёры:*  Авторизацию должны пройти все актеры. |
| *Предусловия:*  Пользователь не авторизован системой. |
| *Основной поток:*   1. Прецедент начинается, когда пользователь запускает приложение. 2. Пользователь вводит данные для входа в систему. 3. Введенные данные проверяются сервером. 4. Если введенные данные корректны, то пользователь авторизован и получает доступ к функциям приложения, иначе пользователю указывается ошибка. |
| *Постусловия:*  Пользователь авторизован. |
| *Альтернативный поток:*   1. Прецедент начинается, когда пользователь запускает приложение. 2. Пользователь выбирает авторизацию через google аккаунт. 3. Введенные данные проверяются сервером. 4. Если введенные данные корректны, то пользователь авторизован и получает доступ к функциям приложения, иначе пользователю указывается ошибка. |

Таблица 1.4 - Прецедент : Создание рабочего пространства

|  |
| --- |
| ID: 3 |
| Краткое описание:  Система позволяет авторизированному пользователю, создать рабочее пространство в котором он может объединить свои проекты. |
| Главные актеры:  Авторизованный пользователь (User). |
| Второстепенные актеры:  Нет. |
| Предусловие:   1. Пользователь вошел в систему. |
| Основной поток:   1. Пользователь выбрал опцию «Создать рабочее пространство». 2. Система выводит форму (окно), в которой необходимо указать название рабочего пространства. 3. Пользователь вводит текст. 4. Пользователь нажимает кнопку применить. 5. Система создает рабочее пространство. |
| Постусловие:  В системе создано рабочее пространство с указанным именем. |
| Альтернативный поток:  Нет. |

Таблица 1.5 - Прецедент: Создание проекта

|  |
| --- |
| ID: 4 |
| Краткое описание:  Пользователь имеет возможность создать проект и добавить его в существующее рабочее пространство |
| Главные актеры:  Пользователь. |
| Второстепенные актеры:  Нет. |
| Предусловие:   1. Пользователь вошел в систему. 2. Создано хотя бы одно рабочее пространство. |
| Основной поток:   1. Прецедент начинается, когда пользователь выбирает пункт «Создание проекта» 2. Пользователь вводит название и описание проекта 3. Пользователь выбирает рабочее пространство и выбирает опцию «Сохранить» 4. Система создает проект с указанным именем и добавляет его в выбранное рабочее пространство |
| Постусловие:  Создан новый проект Пользователя и добавлен в выбранное рабочее пространство |
| Альтернативный поток:  Нет. |

Таблица 1.6 - Прецедент : Создание записи

|  |
| --- |
| ID: 5 |
| Краткое описание:  Система позволяет авторизированному пользователю создать запись (задачу) для ведения учета затраченного на эту запись (задачу) времени.. |
| Главные актеры:  Авторизованный пользователь (User). |
| Второстепенные актеры:  Нет. |
| Предусловие:   1. Пользователь вошел в систему. |
| Основной поток:   1. Пользователь выбрал опцию «Создать запись». 2. Система выводит форму (окно), в которой необходимо указать описание и выбрать проект. 3. Пользователь вводит текст описания. 4. Пользователь выбирает проект. 5. Пользователь нажимает кнопку сохранить. 6. Система создает запись с указанным описанием в выбранном проекте. |
| Постусловие:  В системе создана запись с указанным описанием в выбранном пользователем проекте. |
| Альтернативный поток:  Нет |

Таблица 1.7 - Прецедент : Ввод затраченного времени

|  |
| --- |
| ID: 6 |
| Краткое описание:  Система позволяет авторизированному пользователю вести учет затраченного времени. |
| Главные актеры:  Авторизованный пользователь (User). |
| Второстепенные актеры:  Время. |
| Предусловие:   1. Пользователь вошел в систему. 2. Пользователь выбрал запись (задачу). |
| Основной поток:   1. Пользователь выбрал опцию «Ввод затраченного времени». 2. Система выводит форму (окно), в которой необходимо указать способ ввода затраченного времени: ручной режим или автоматизированный режим. 3. Пользователь выбирает ручной режим. 4. Пользователь задает начало и конец отсчета. 5. Система добавляет к выбранной записи значение затраченного времени. |
| Постусловие:  В выбранную запись (задачу) добавлено значение затраченного времени. |

|  |
| --- |
| Альтернативный поток:   1. Пользователь выбрал опцию «Ввод затраченного времени». 2. Система выводит форму (окно), в которой необходимо указать способ ввода затраченного времени: ручной режим или автоматизированный режим. 3. Пользователь выбирает автоматизированный режим. 4. Пользователь нажимает на кнопку «Начать отсчет». 5. Через некоторый промежуток времени пользователь нажимает на кнопку «Закончить отсчет». 6. Система добавляет к выбранной записи значение затраченного времени. |
|  |

# Анализ

## Классы анализа

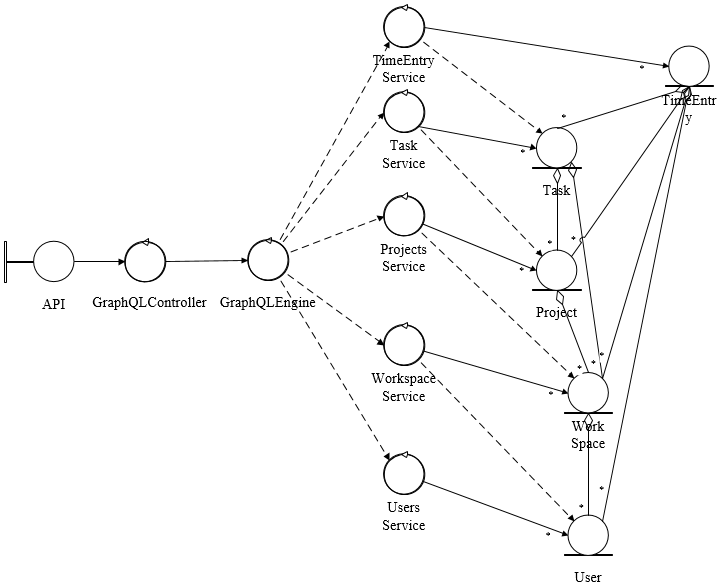


Рисунок 2.1 – Диаграмма классов анализа

Диаграмма классов анализа содержит набор объектов модели приложения, которыми управляют соответствующие сервисы.

Все запросы, которые могут быть обработаны серверным приложением отправляются на GraphQLController. В нем из тела http запроса извлекаются отправленные данные и формируется обращение к механизмам GraphQL технологии.

GraphQLEngine (GraphQL) – это библиотека позволяющая использовать специальный синтаксис в http запросах, который описывает как запрашивать данные, и, в основном, используется клиентом для загрузки данных с сервера.

GraphQL имеет три основные характеристики:

1. Позволяет клиенту точно указать, какие данные ему нужны;
2. Облегчает агрегацию данных из нескольких источников;
3. Использует систему типов для описания данных.

Аналогично чтению данных, GraphQL может использоваться и для записи новой информации на серверное приложение. Функции, отвечающие за это, называются: мутации. Все типы, мутации и функции чтения данных указываются в схемах.

GraphQLEngine скрывает в себе реализации парсинга схем, согласования функций описанных в схемах с реальными обработчиками серверного приложения и сами обработчики.

Workspace – сущность имеет смысл рабочего места и объединяет в себе связанные по смыслу объекты типа Project. Имеет имя.

Project – сущность объединяющая в себе связанные по смыслу задачи. Содержит только имя и ссылки на сущности типа Task.

Task – сущность отражающая некоторую деятельность. Может содержать ряд временных промежутков, описание и имя.

TimeEntry – сущность по работе со временем. Содержит время старта, окончания и длительность.

Workspace Service – отвечает за все функции необходимые при работе с типом Workspace. А также управляет временем жизни объектов этого типа.

Project Service – отвечает за все функции необходимые при работе с типом Project. А также управляет временем жизни объектов этого типа.

Task Service – отвечает за все функции необходимые при работе с типом Task. А также управляет временем жизни объектов этого типа.

TimeEntry Service – отвечает за все функции необходимые при работе с типом TimeEntry. А также управляет временем жизни объектов этого типа.

User Service – отвечает за все функции необходимые при работе с типом User. А также управляет временем жизни объектов этого типа.

## Диаграмма последовательностей классов анализа

ОПИСАТЬ

## Диаграммы деятельности

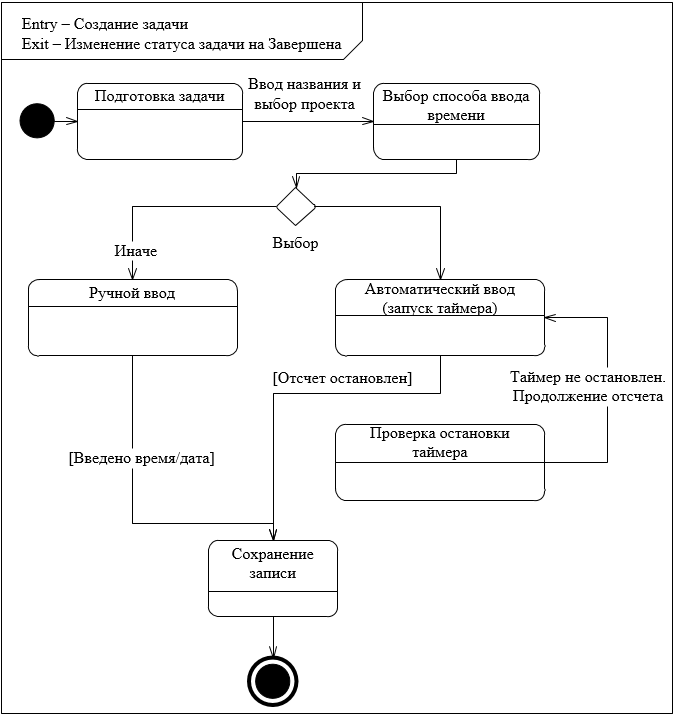


Рисунок 2.2 – Диаграмма деятельности для «Создания задачи»

## Конечные автоматы для динамических классов анализа

СДЕЛАТЬ КОНЕЧНЫЙ АВТОМАТ

## Пакеты системы

## 

# Проектирование

## Проектные классы

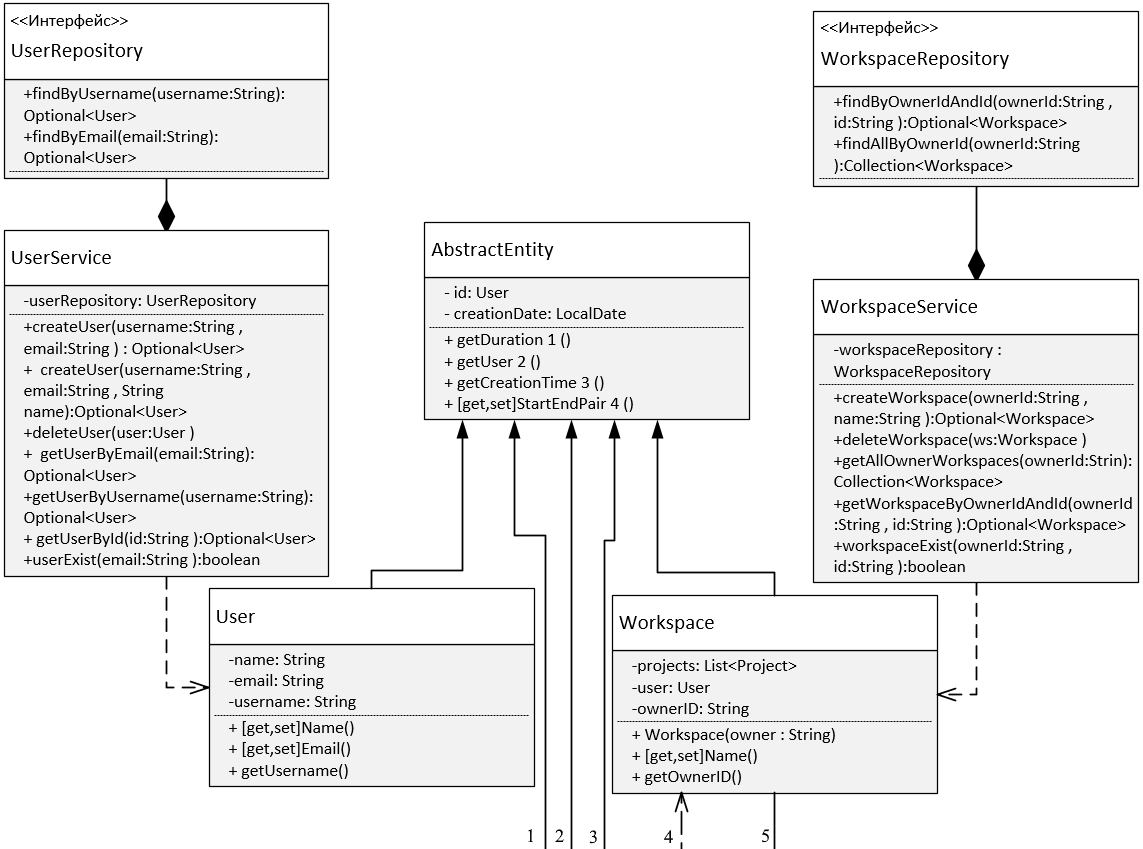


Рисунок 3.1 - Диаграмма проектных классов. Часть 1

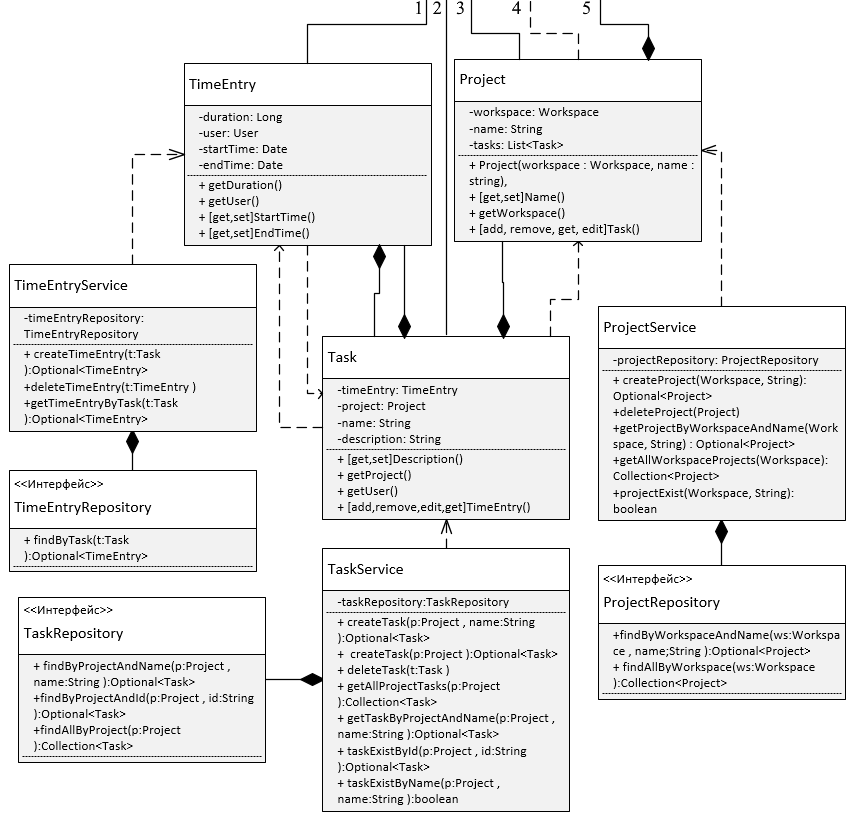


Рисунок 3.2 - Диаграмма проектных классов. Часть 2

## Диаграммы последовательностей для операций проектных классов

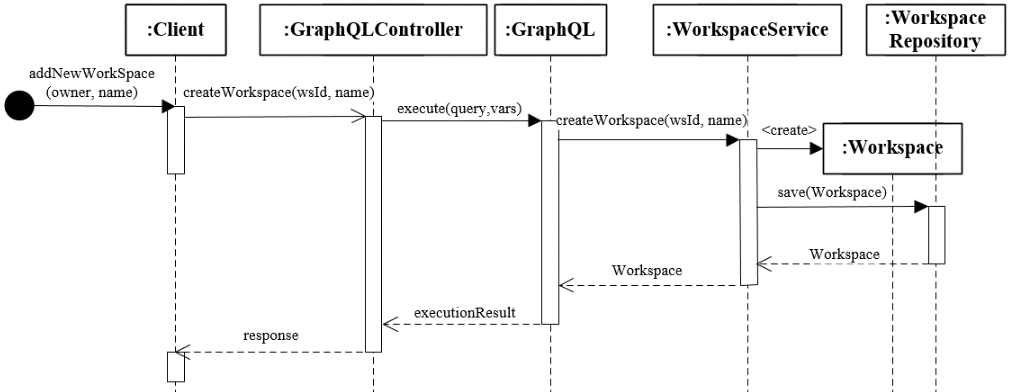


Рисунок 3.3 – Диаграмма последовательности создания рабочего пространства

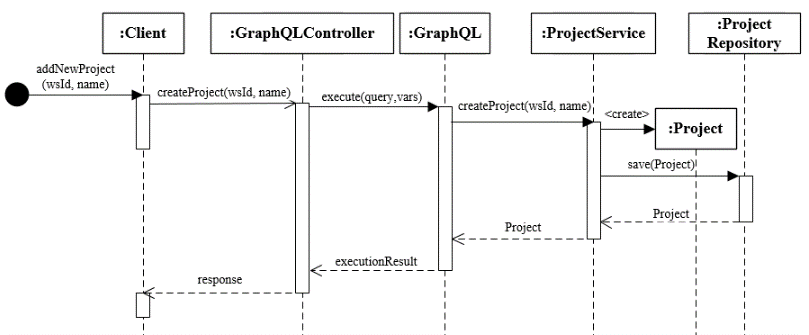


Рисунок 3.4 – Диаграмма последовательности создания проекта

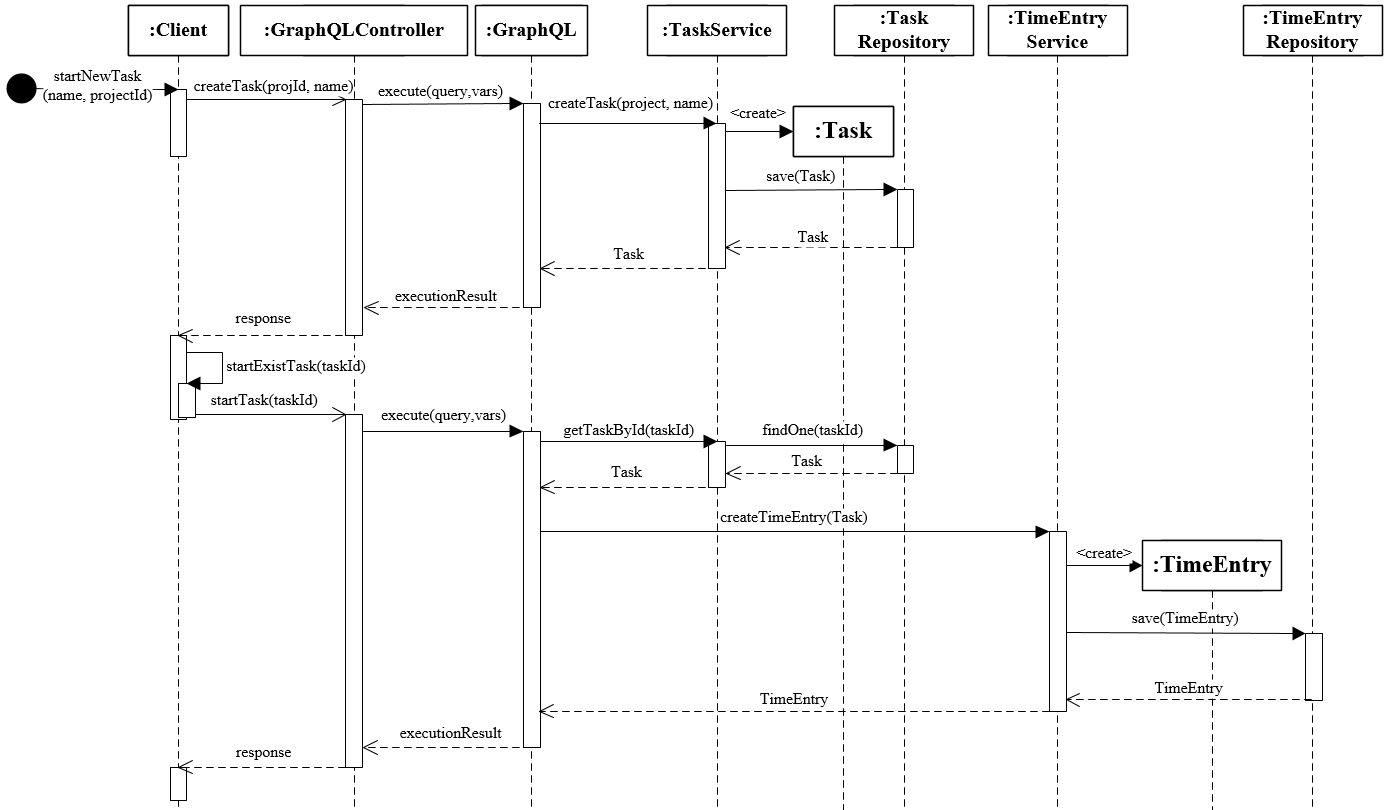


Рисунок 3.5 - Диаграмма последовательности создания и запуска задачи

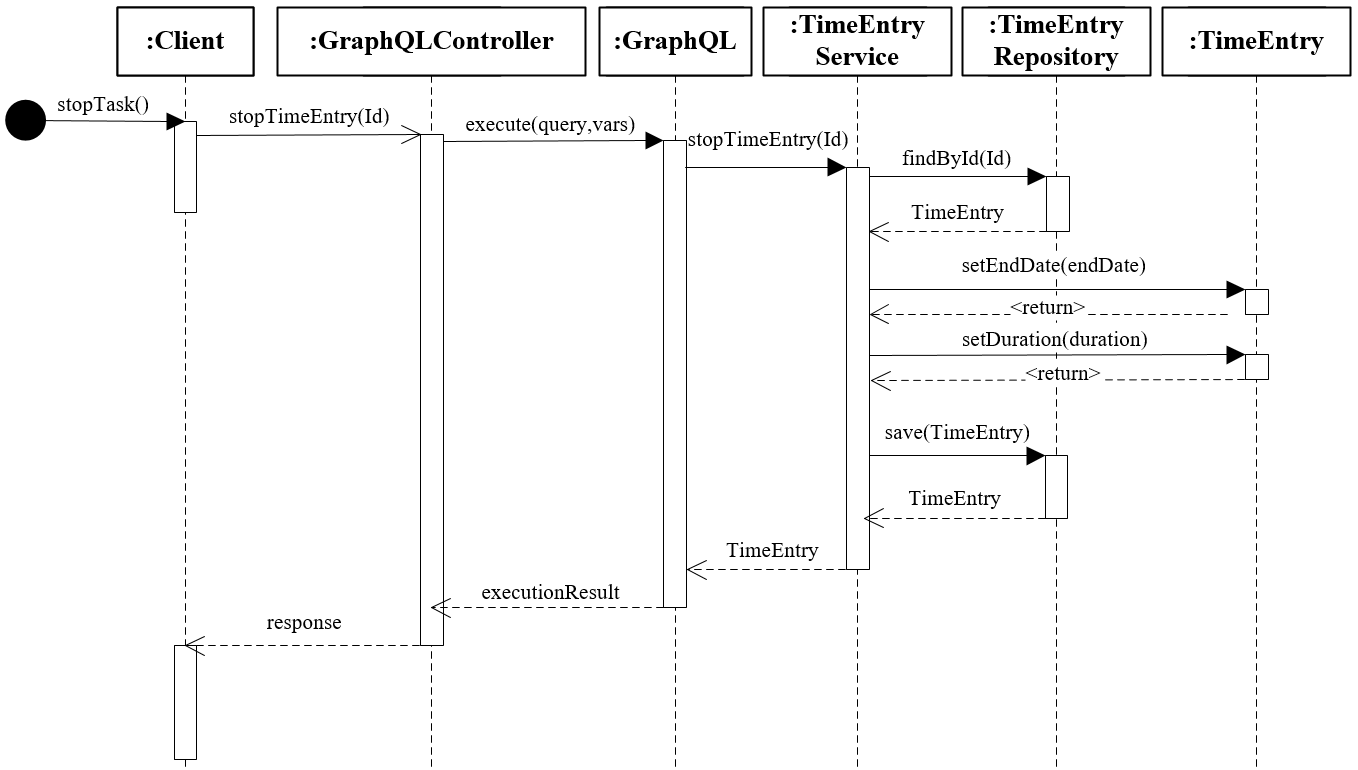


Рисунок 3.6 – Диаграмма последовательности завершения задачи

# Реализация

## Интеграционное тестирование системы

## Непрерывная интеграция

Непрерывная интеграция – это практика разработки программного обеспечения, которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем. Непрерывная интеграция является одним из основных приемов экстремального программирования.

В качестве систем непрерывной интеграции выбраны веб-сервисы Travis CI, AppVeyor.

### Travis CI

Travis CI – распределенный веб-сервис для сборки и тестирования программного обеспечения, использующий GitHub в качестве хостинга исходного кода. Данный сервис бесплатен для публичных проектов. В курсовом проекте Travis CI используется для непрерывной интеграции проекта серверного ПО, а также проекта клиентского ПО, разработанного под Android платформу.

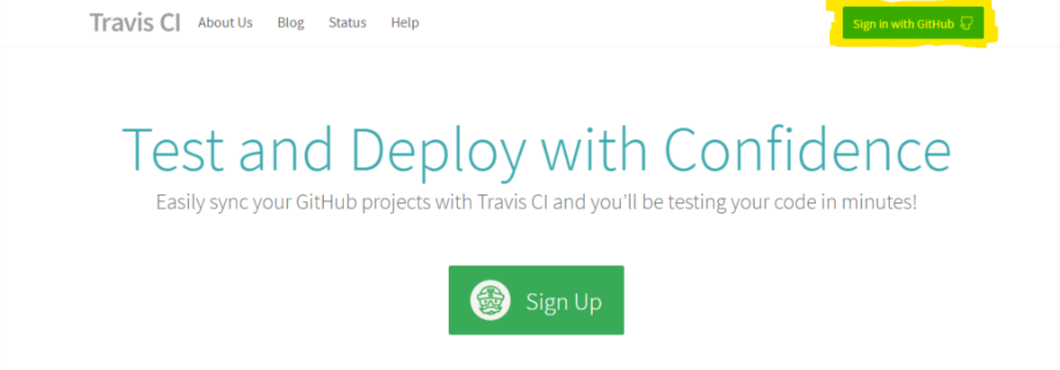


Рисунок 4.1

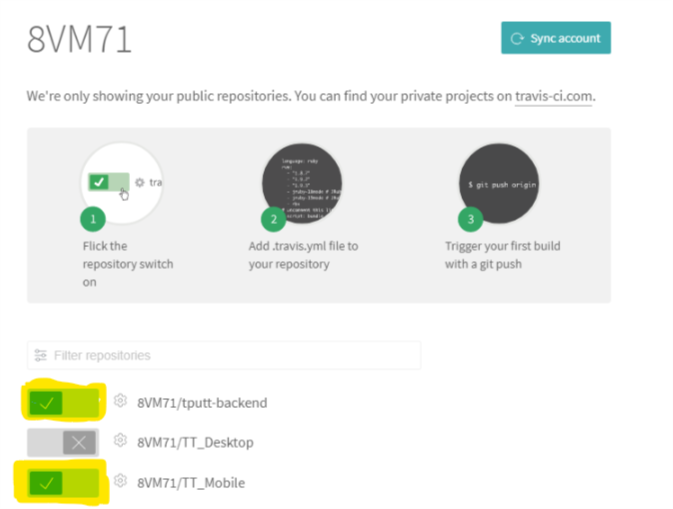


Рисунок 4.2

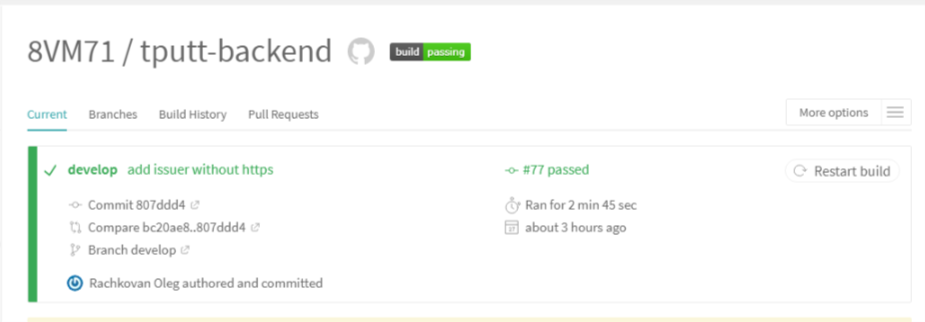


Рисунок 4.3



Рисунок 4.4

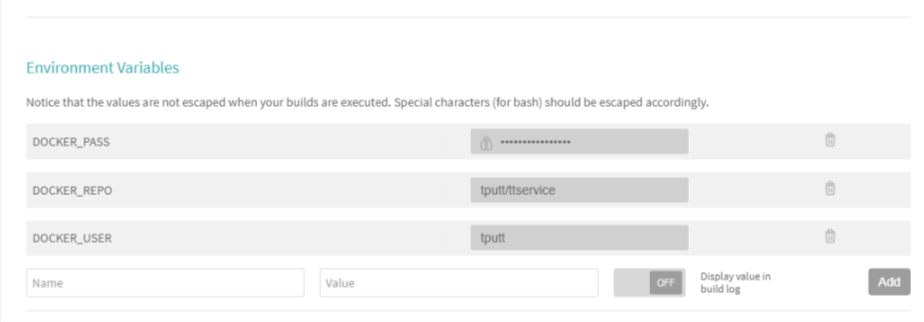


Рисунок 4.5

### AppVeyor

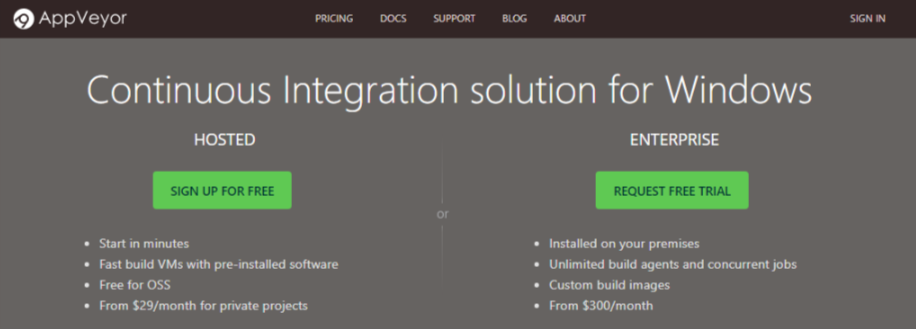


Рисунок 4.6

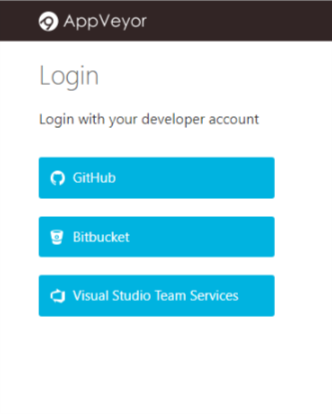


Рисунок 4.7

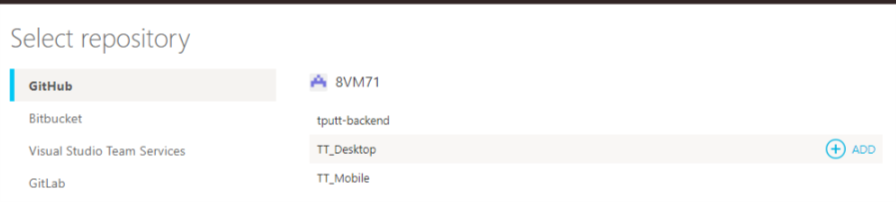


Рисунок 4.8

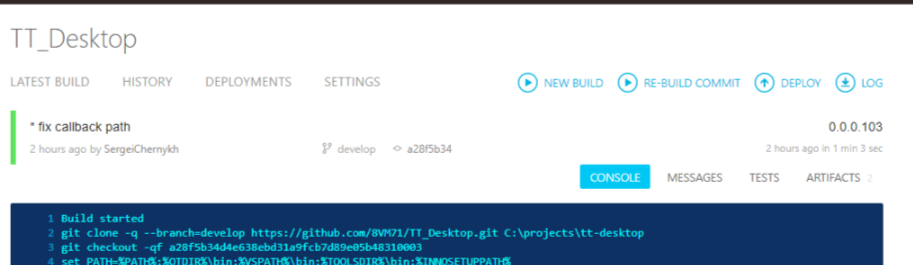


Рисунок 4.9

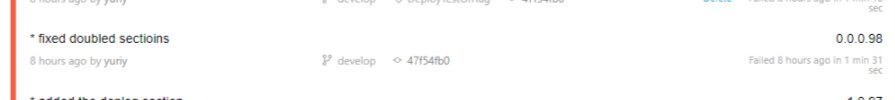


Рисунок 4.10

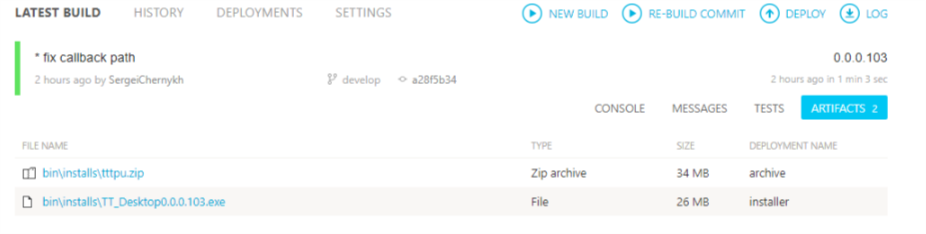


Рисунок 4.11

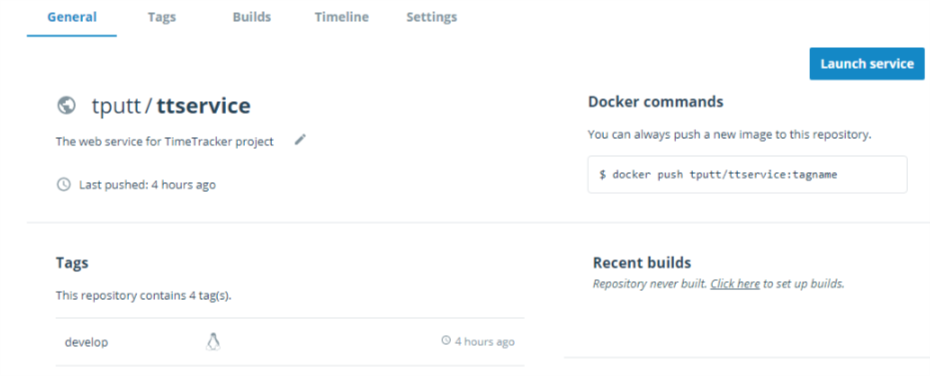


Рисунок 4.12

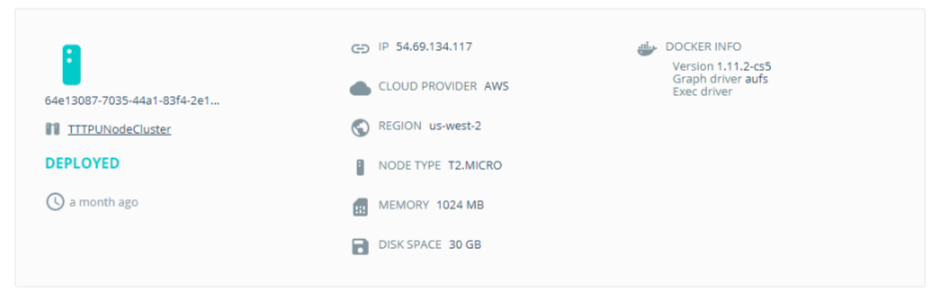


Рисунок 4.13

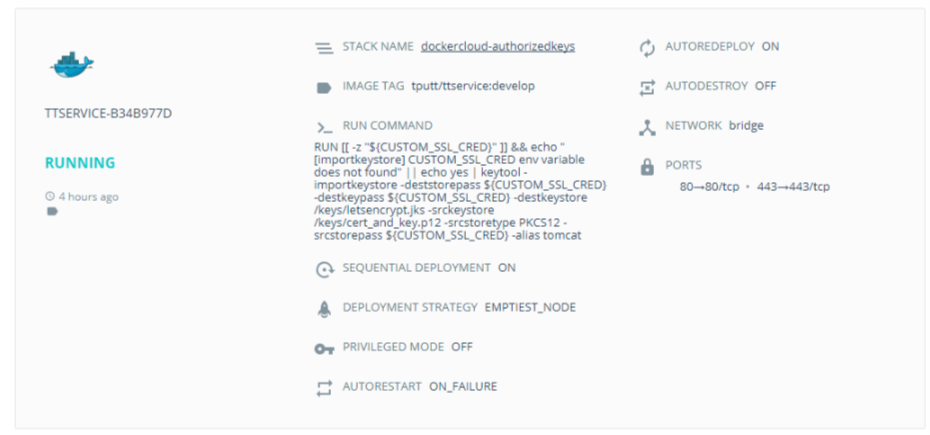


Рисунок 4.14

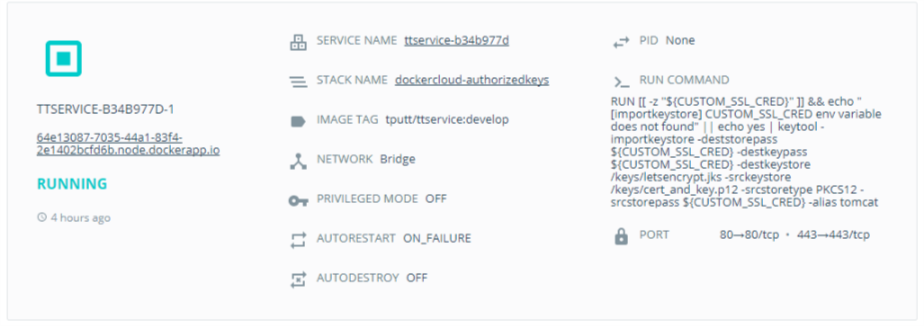


Рисунок 4.15

# Документация

# Заключение

# Список использованных источников

2. Прата, Стивен. - Язык программирования C++. Лекции и упражнения, 6-е изд.: Пер. с англ. — М.: ООО "И.Д. Вильяме", 2012. - 1248 с.
3. Graphql.org
4. Spring.io
5. Github.com
6. Developers.google.com
7. docs.travis-ci.com
8. cloud.docker.com
9. aws.amazon.com
10. appveyor.com/docs
11. doc.qt.io
12. letsencrypt.org
13. developer.android.com
14. gradle.org
15. docs.spockframework.org
16. docs.docker.com
17. jrsoftware.org