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](#_Toc502305319)

[1 Требования к программе 4](#_Toc502305320)

[1.1 Назначение программы 4](#_Toc502305321)

[1.2 Область применения 4](#_Toc502305322)

[1.3 Варианты использования 4](#_Toc502305323)

[2 Анализ 13](#_Toc502305324)

[2.1 Классы анализа 13](#_Toc502305325)

[2.2 Диаграммы деятельности 15](#_Toc502305326)

[2.3 Пакеты системы 16](#_Toc502305327)

[3 Проектирование 17](#_Toc502305328)

[3.1 Проектные классы 17](#_Toc502305329)

[3.2 Диаграммы последовательностей для операций проектных классов 19](#_Toc502305330)

[4 Реализация 22](#_Toc502305331)

[4.1 Интеграционное тестирование системы 22](#_Toc502305332)

[4.2 Непрерывная интеграция 23](#_Toc502305333)

[4.2.1 Travis CI 23](#_Toc502305334)

[4.2.2 AppVeyor 26](#_Toc502305335)

[5 Документация 33](#_Toc502305336)

[Заключение 34](#_Toc502305337)

[Список использованных источников 35](#_Toc502305338)

# Введение

В жёстком мире капитализма, где время — это деньги и нет ничего личного, работодатель не хочет платить за недоработку, а сотрудник скрупулёзно считает каждый предназначенный ему доллар.

Для оценки личной эффективности, эффективности персонала, для получения полной картины того, как мы тратим свое время, широко применяется программное обеспечение классов time tracking software (программное обеспечение автоматизированного учета времени), task management software (сервис управления прое), системы персональной продуктивности (PPS), системы GTD.

# Требования к программе

## Назначение программы

## Область применения

## Варианты использования

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

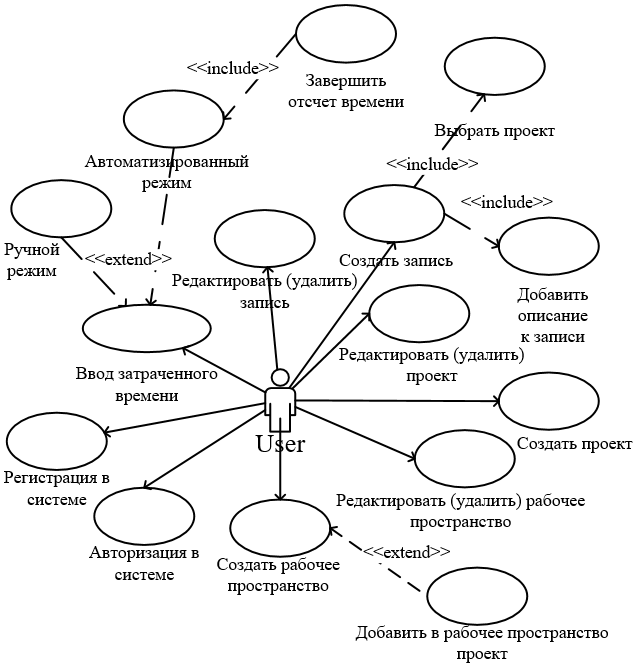


Рисунок 1.1 – Диаграмма вариантов использования

Данная диаграмма вариантов использования (Рисунок 1.1) отображает доступную функциональность серверного приложения. Среди которых есть как простые операции создание\обновление\удаление существующих сущностей, так и сложные: авторизация, отсчет времени В настоящий момент вся функциональность, описанная на диаграмме, кроме регистрации в системе, реализована. Подробное описание вариантов использования приведены в таблицах 1.2 - 1.6.

Таблица 1.1 - Варианты использования

|  |  |
| --- | --- |
| Регистрация в системе | - |
| Авторизация в системе | Авторизация в системе использую учетные данные google аккаунта. |
| Создание рабочего пространства | Создание сущности типа Workspace |
| Редактирование (удаление) рабочего пространства | Обновление полей существующего объекта или удаление оного. |
| Создание проекта | Создание сущности типа Project |
| Редактирование (удаление) проекта | Обновление полей существующего объекта или удаление оного. |
| Создание задания | Создание сущности типа Task |
| Редактирование (удаление) записи | Обновление полей существующего объекта или удаление оного. |
| Ввод рабочего времени | Время вводится либо в ручном режиме, либо автоматически при создании задания. |

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

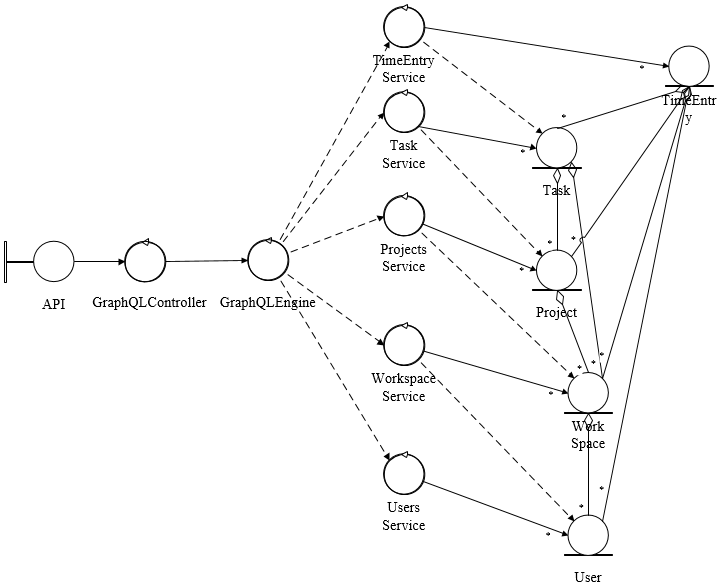


Рисунок 2.1 – Диаграмма классов анализа

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.

## Диаграммы деятельности

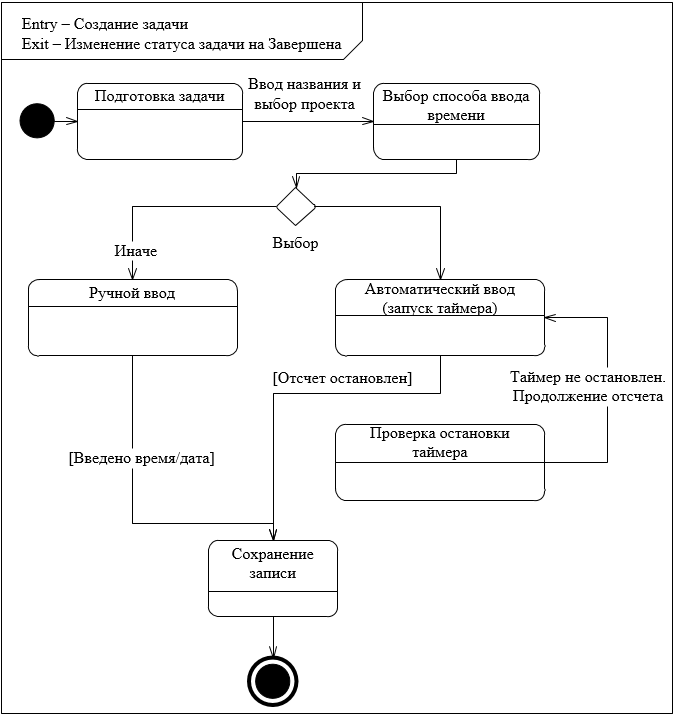


Рисунок 2.2 – Диаграмма деятельности для «Создания задачи»

## Пакеты системы

## 

# Проектирование

## Проектные классы

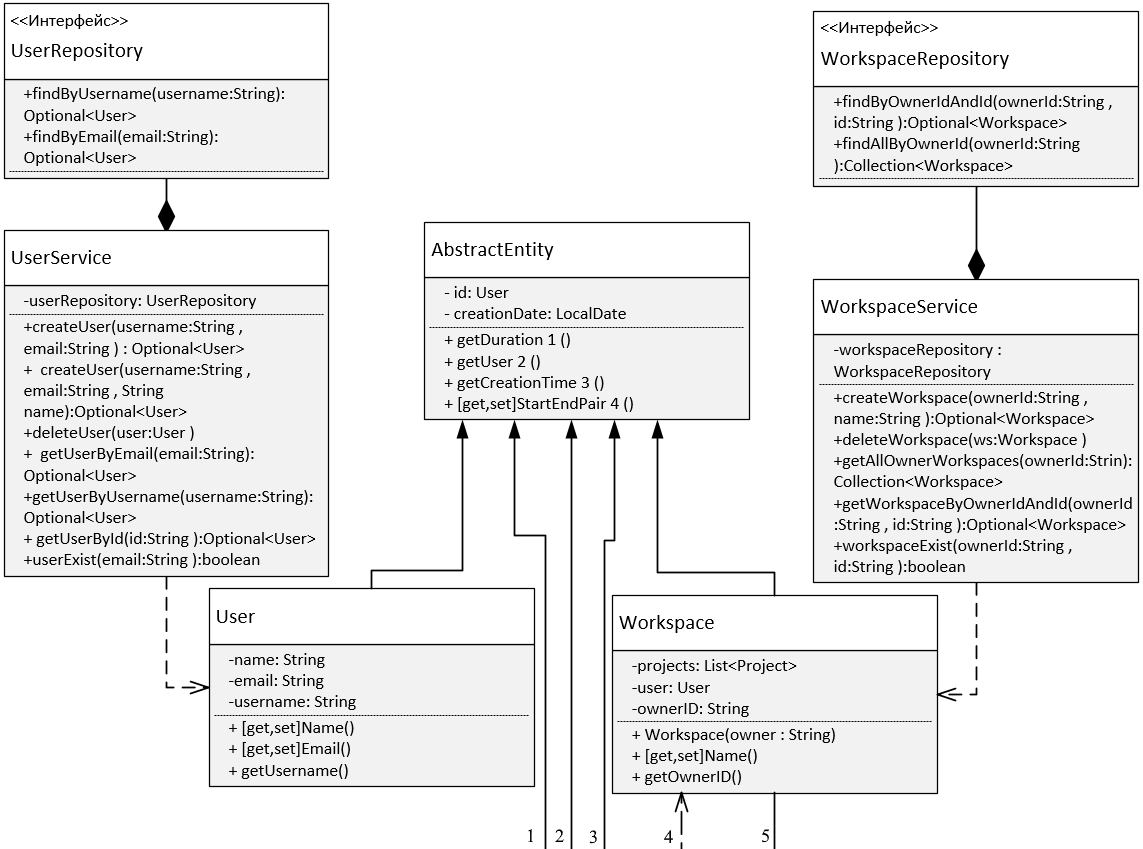


Рисунок 3.1 - Диаграмма проектных классов. Часть 1

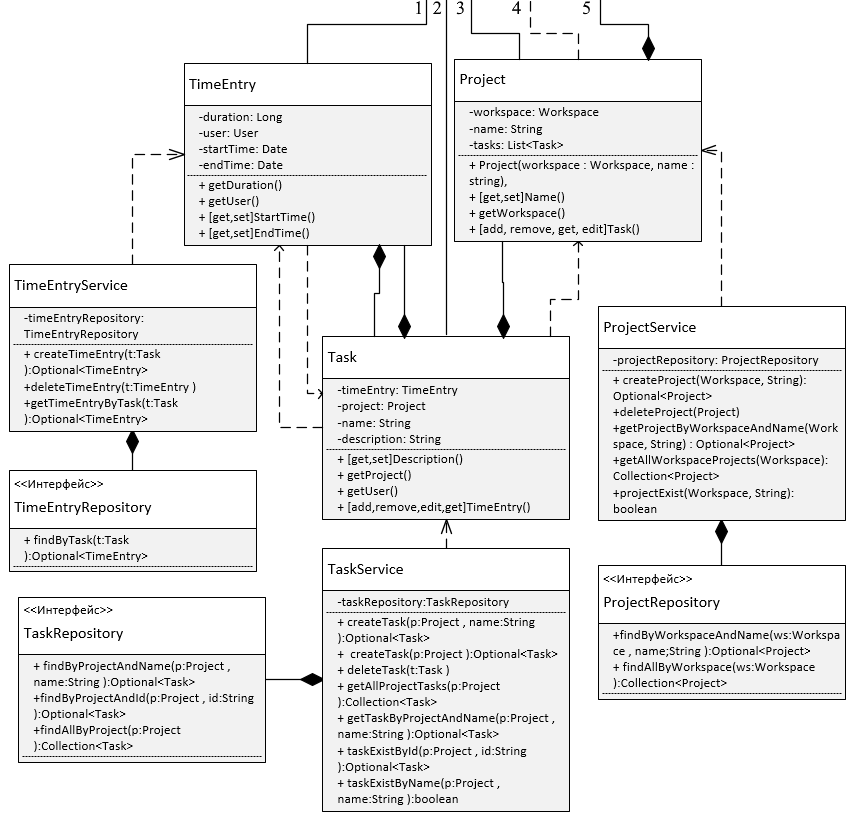


Рисунок 3.2 - Диаграмма проектных классов. Часть 2

## Диаграммы последовательностей для операций проектных классов

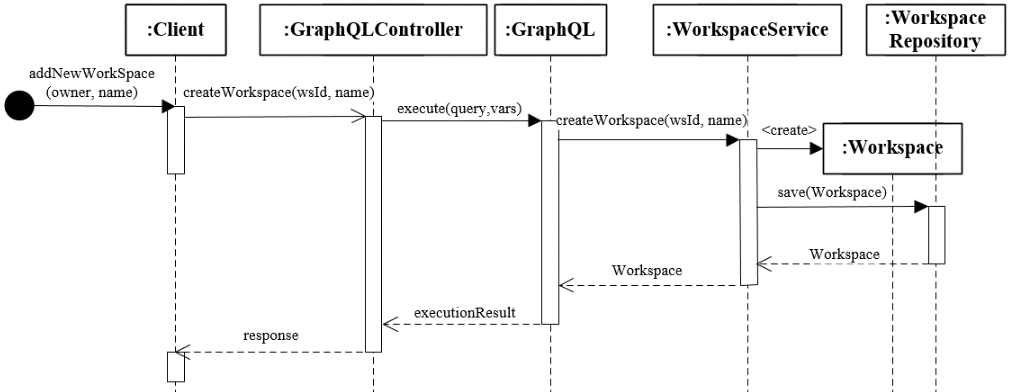


Рисунок 3.3 – Диаграмма последовательности создания рабочего пространства

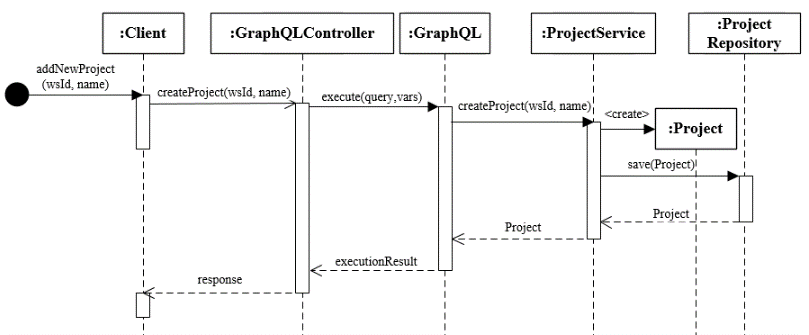


Рисунок 3.4 – Диаграмма последовательности создания проекта

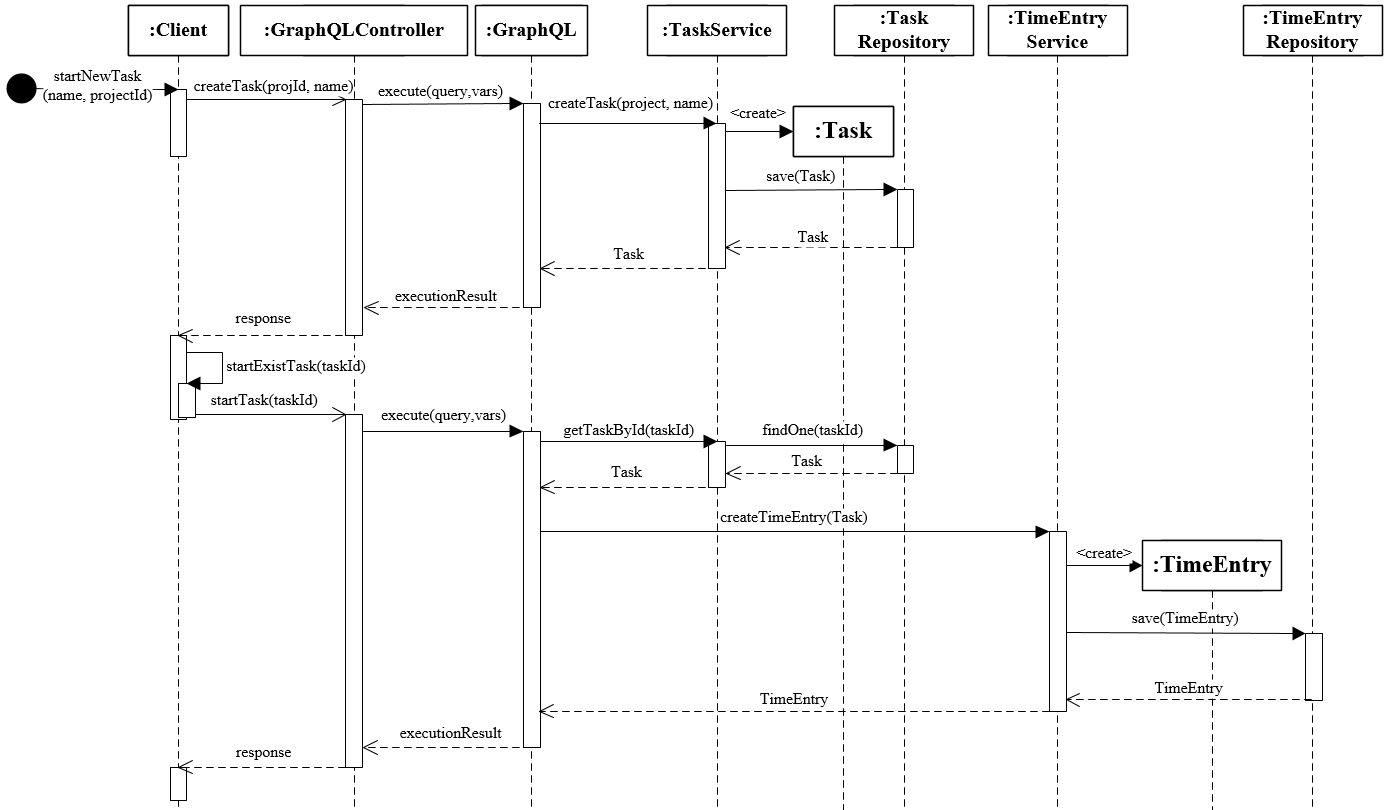


Рисунок 3.5 - Диаграмма последовательности создания и запуска задачи

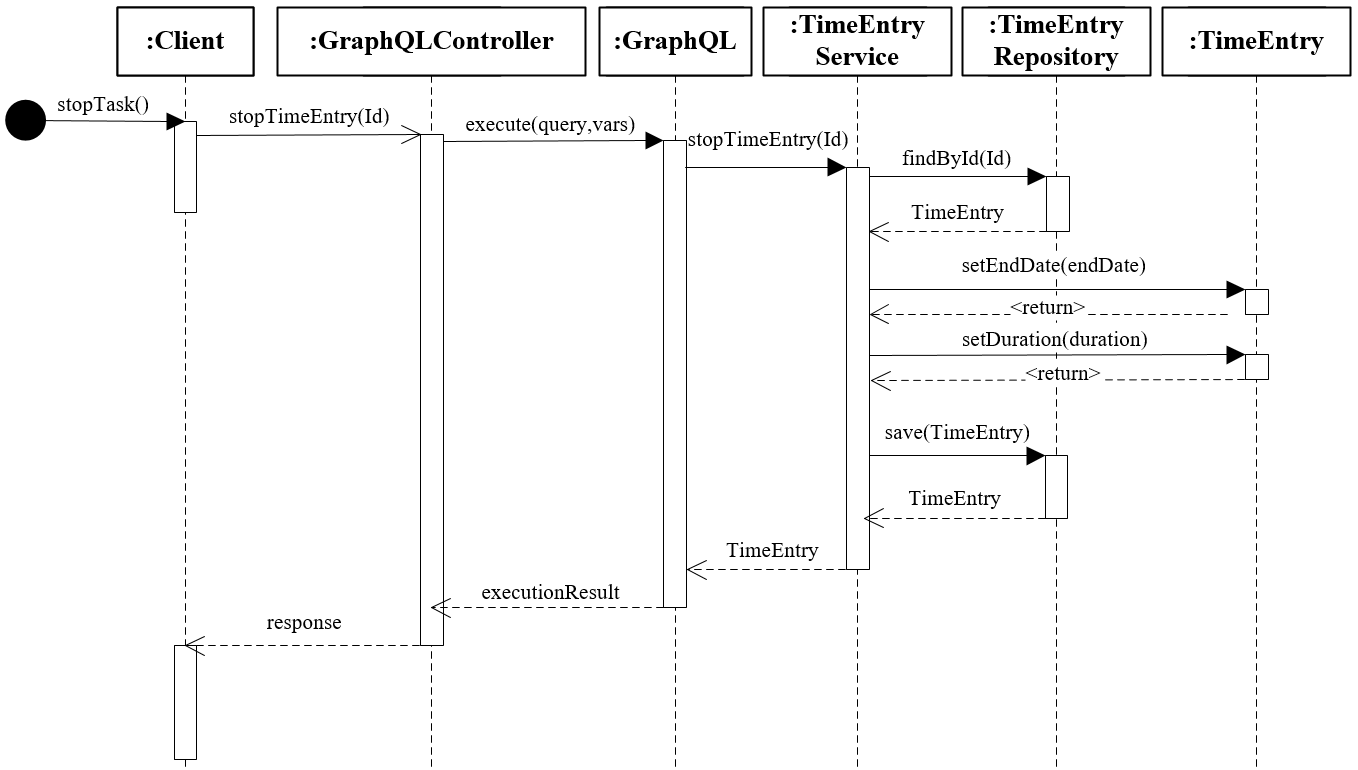


Рисунок 3.6 – Диаграмма последовательности завершения задачи

# 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.

Первым шагом является настройки системы является вход в учетную запись Travis CI через аккаунт на GitHub. Для этого необходимо на странице <https://travis-ci.org> нажать на кнопку “Sign In With Github” (Рисунок 4.1).

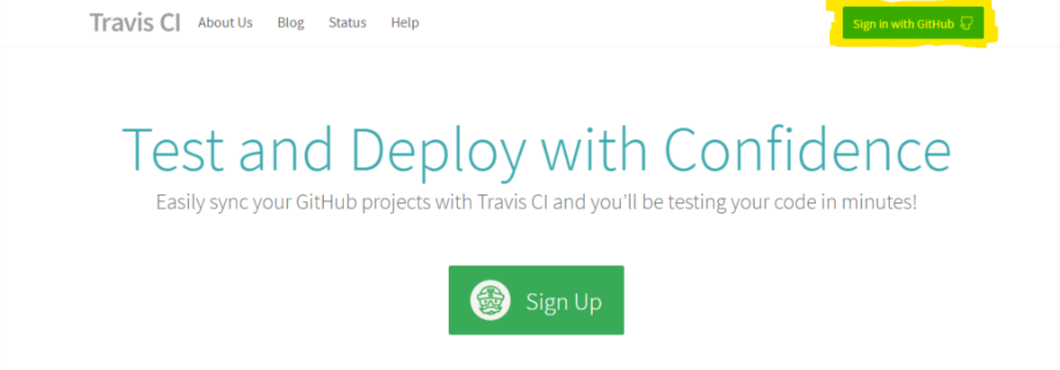


Рисунок 4.1- Главная страница Travis CI.

После входа на сервис необходимо активировать синхронизацию репозитория проекта на GitHub с сервисом. Для это необходимо зайти в настройки профиля и привести необходимые переключатели во включенное положение как показано на Рисунок 4.2.

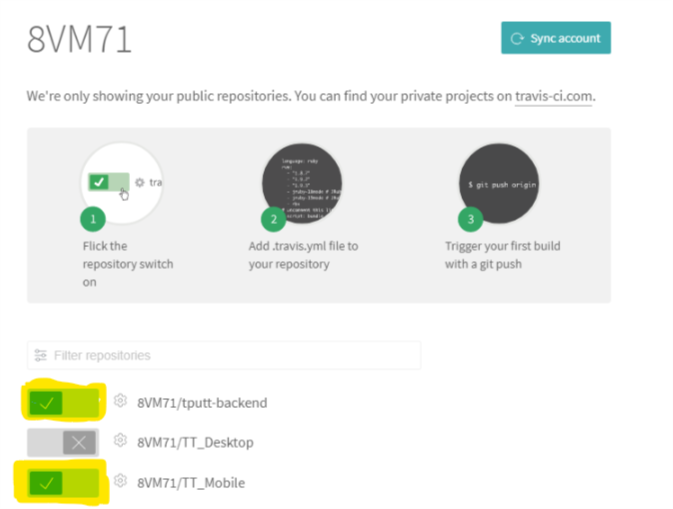


Рисунок 4.2 – Настройка доступа Trais CI к репозиторию

Следующим шагом является добавление в корневую директорию проекта файла .travis.yml. В этом файле содержится скрипт, в котором определены настройки окружения сборки проекта, скрипты, выполняемые перед сборки, скрипты сборки и скрипты, выполняемые по окончанию сборки. Всю подробную информацию о настройке данного файла можно найти на сайте Travis CI.

Содержание файла .travis.yml для проекта серверного ПО представлено ниже:

sudo: required

services:

- docker

language: java

jdk:

- oraclejdk8

before\_install:

- chmod +x gradlew

install: true

cache:

directories:

- $HOME/.gradle/caches/

- $HOME/.gradle/wrapper/

script:

- ./gradlew clean

- ./gradlew test --rerun-tasks

- ./gradlew assemble

after\_success:

- sh .travis/deploy\_dockerhub.sh

- if [ -e ./gradlew ]; then ./gradlew jacocoTestReport;else gradle jacocoTestReport;fi

- bash <(curl -s <https://codecov.io/bash>)

В данном файле указывается, что проект написан на языке Java. На основе этой информации Travis автоматически подготовит необходимое окружение для сборки, характерное для сборки большинства проектов на этом языке. В секции script указывается скрипт сборки приложения. Во время сборки приложения автоматически соберутся и запустятся unit-тесты и если некоторые из них не пройдут, то сборка будет считаться неудачной. После удачной сборки проекта выполняется скрипт сборки Docker образа, содержащем ПО и отправка этого образа в Docker репозиторий. После выполняется отправка отчета о покрытии кода тестами на сервис CodeCov, где анализируется процент покрытия кода тестами.

Ниже представлен рисунок успешно выполненной сборки (Рисунок 4.3).

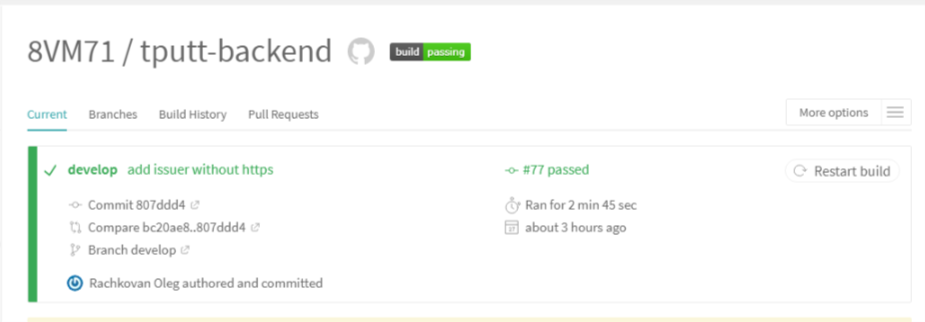


Рисунок 4.3 – Сборка успешно выполнена

Результат неудачной сборки (Рисунок 4.4)



Рисунок 4.4 – Сборка окончена со статусом “failed”

### AppVeyor

Для настольного приложения в качестве системы непрерывной интеграции была выбрана система AppVeyor. Данный сервис специализируется на работе с Windows приложениями. Также он бесплатен для открытых проектов GitHub.

Для настройки сначала необходимо войти в учетную запись AppVeyor через аккаунт GitHub (Рисунок 4.5-Рисунок 4.6).

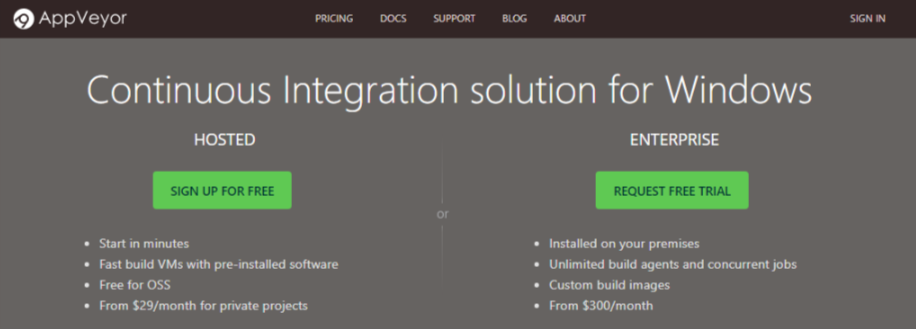


Рисунок 4.5 – Главная страница AppVeyor

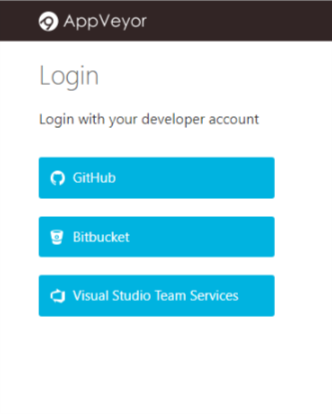


Рисунок 4.6 – Форма входа

После на вкладке Projeсts выбрать пункт New Project и выбрать необходимый репозиторий проекта (Рисунок 4.7).

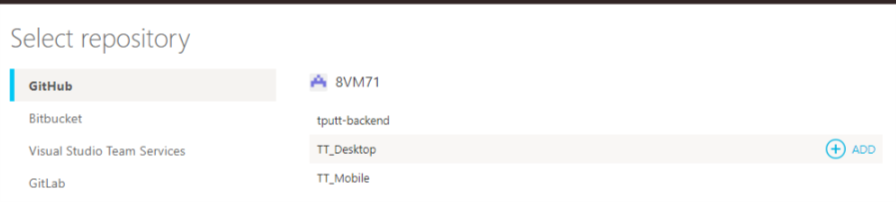


Рисунок 4.7 – Добавление проекта

После добавления проекта в сервис AppVeyor нужно в корневую директорию проекта добавить файл appveyor.yml, где описать конфигурацию процесса сборки.  
Код файла для проекта настольного клиента приведен ниже:

version: 0.0.0.{build}

image: Visual Studio 2015

branches:

only:

- develop

skip\_tags: true

environment:

QTDIR: C:\Qt\5.9.2\msvc2015

VSPATH: C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC

TOOLSDIR: C:\Qt\Tools\QtCreator

INNOSETUPPATH: C:\Program Files (x86)\Inno Setup 5\

INSTALLER\_NAME: "%APPVEYOR\_PROJECT\_NAME%%APPVEYOR\_BUILD\_VERSION%"

PR\_NAME: "%APPVEYOR\_PROJECT\_NAME%"

BUILD\_V: "%APPVEYOR\_BUILD\_VERSION%"

BUILD\_F: "%APPVEYOR\_BUILD\_FOLDER%"

before\_build:

- set PATH=%PATH%;%QTDIR%\bin;%VSPATH%\bin;%TOOLSDIR%\bin;%INNOSETUPPATH%

- call "%VSPATH%\vcvarsall.bat"

- mkdir build

build\_script:

- cd build

- qmake ../tputt.pro -spec win32-msvc

- jom

after\_build:

- windeployqt --release %APPVEYOR\_BUILD\_FOLDER%\bin\release --qmldir %APPVEYOR\_BUILD\_FOLDER%

- xcopy C:\OpenSSL-Win32\bin\libeay32.dll %APPVEYOR\_BUILD\_FOLDER%\bin\release /y

- xcopy C:\OpenSSL-Win32\bin\ssleay32.dll %APPVEYOR\_BUILD\_FOLDER%\bin\release /y

- iscc.exe /Qp /DAPPVEYOR\_PROJECT\_NAME=%PR\_NAME% /DAPPVEYOR\_BUILD\_VERSION=%BUILD\_V% /DINSTALLER\_NAME=%INSTALLER\_NAME% /DAPPVEYOR\_BUILD\_FOLDER=%BUILD\_F% "%APPVEYOR\_BUILD\_FOLDER%\.appveyor\installer.iss"

- 7z a -y ..\bin\installs\tttpu.zip %APPVEYOR\_BUILD\_FOLDER%\bin\release\\*

artifacts:

- path: bin/installs/tttpu.zip

name: archive

type: zip

- path: bin/installs/%INSTALLER\_NAME%.exe

name: installer

deploy:

- provider: GitHub

name: production

release: $(appveyor\_project\_name)-v$(appveyor\_build\_version)

description: 'Auto release on master'

artifact: installer

prerelease: true

draft: true

auth\_token:

secure: +I1mbAzf5mFFmAHuGk/fgqd2RWhOQZzen5MRGXYyqZ+dFi5Px2JsPOj/4BGQ4Wl8

on:

branch: develop

В секции *environment* определяются переменные среды, необходимые для процесса сборки. В секции *build\_script* определен скрипт сборки приложения. После успешной сборки приложения выполняется секция *after\_build,* в которой определен скрипт копирования всех необходимых зависимостей, динамически подключаемых библиотек, сборки установочного пакета, а так zip-архива. Секция *artifacts* определяет артефакты сборки, их имена и пути до них (Рисунок 4.10). В секции *deploy* описана конфигурация развертывания установочного пакета на GitHub Releases.

Результат успешной сборки приложения представлена на Рисунок 4.8.

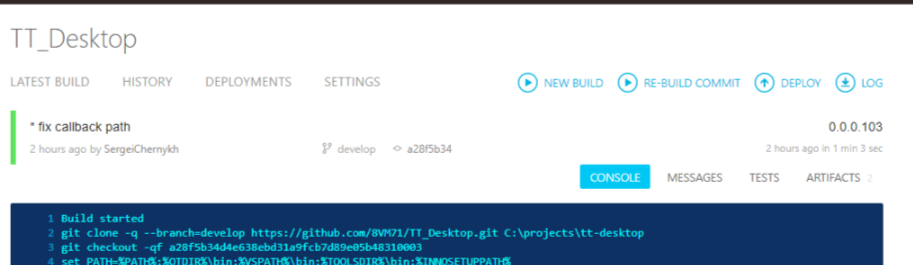


Рисунок 4.8 – Сборка успешно завершена

Результат неудачной сборки (Рисунок 4.9)

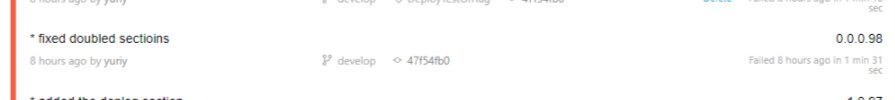


Рисунок 4.9 – Процесс сборки завершился неудачно

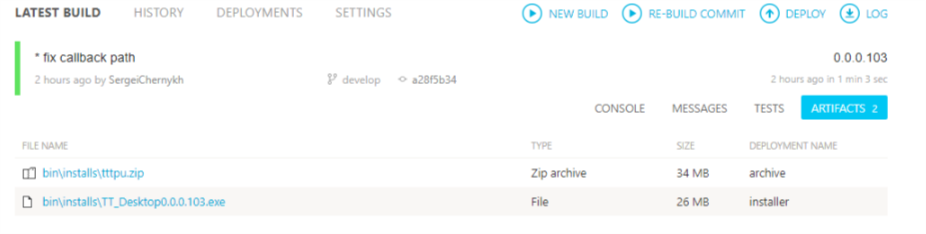


Рисунок 4.10 – Артефакты собранного приложения

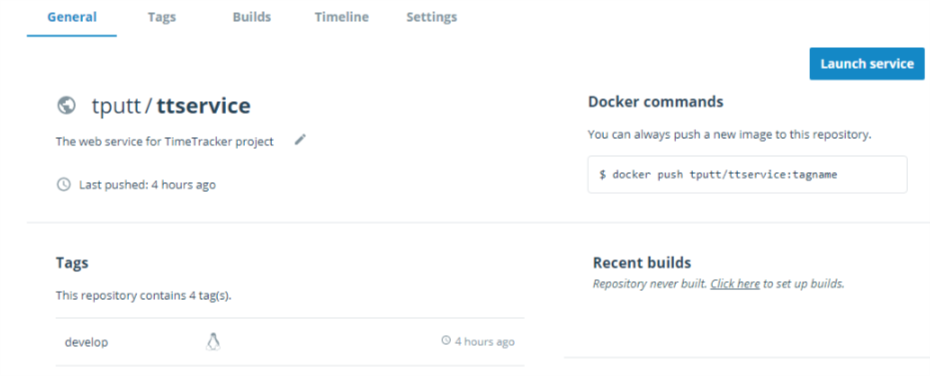


Рисунок 4.11

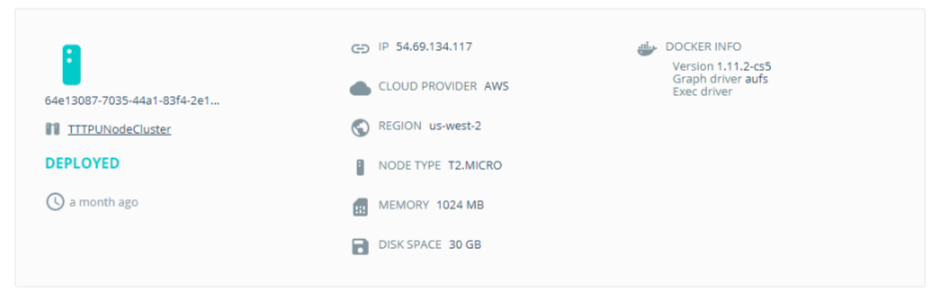


Рисунок 4.12

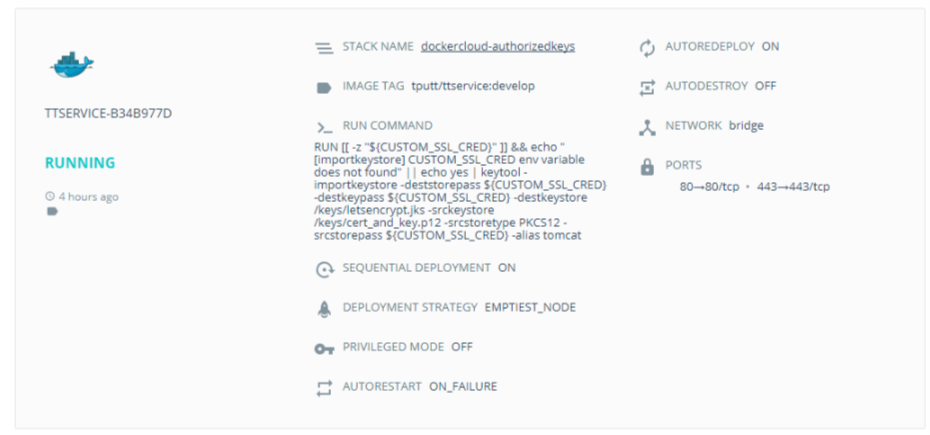


Рисунок 4.13

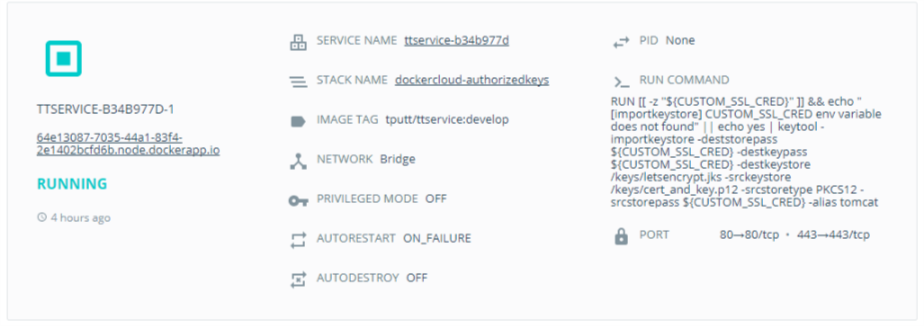


Рисунок 4.14

# Документация

# Заключение

# Список использованных источников

1. Арлоу Д., Нейштадт И. UML 2 и Унифицированный процесс. Практический объектно­ориентирован­ ный анализ и проектирование, 2­е издание. – Пер. с англ. – СПб: Символ­ Плюс, 2007. – 624 с., ил;
2. A query language for your API [Электронный ресурс] // Режим доступа: http://graphql.org/ (дата обращения 29.12.2017);
3. Spring Framework 5.0 // Режим доступа: http://spring.io (дата обращения 29.12.2017);
4. Built for developers [Электронный ресурс] // Режим доступа: http://github.com (дата обращения 29.12.2017);
5. Google Developers Docs [Электронный ресурс] // Режим доступа: http://developers.google.com (дата обращения 29.12.2017);
6. Travis CI Guide [Электронный ресурс] // Режим доступа: http://docs.travis-ci.com (дата обращения 29.12.2017);
7. Docker Cloud^ The official cloud service for continuously delivering Docker applications. [Электронный ресурс] // Режим доступа: http://cloud.docker.com (дата обращения 29.12.2017);
8. Amazon Web Services [Электронный ресурс] // Режим доступа: http://aws.amazon.com (дата обращения 29.12.2017);
9. Continuous Integration and Deployment service for Windows [Электронный ресурс] // Режим доступа: http://appveyor.com/docs (дата обращения 29.12.2017);
10. Qt Framework Documentation [Электронный ресурс] // Режим доступа: http://doc.qt.io (дата обращения 29.12.2017);
11. Let's Encrypt - Free SSL/TLS Certificates [Электронный ресурс] // Режим доступа: http://letsencrypt.org (дата обращения 29.12.2017);
12. Android Developers Guide [Электронный ресурс] // Режим доступа: http://developer.android.com (дата обращения 29.12.2017);
13. Gradle Build Tool Guide [Электронный ресурс] // Режим доступа: http://gradle.org /docs/ (дата обращения 29.12.2017);
14. Spock Framework Reference Documentation [Электронный ресурс] // Режим доступа: http://docs.spockframework.org (дата обращения 29.12.2017);
15. Docker Documentation [Электронный ресурс] // Режим доступа: http://docs.docker.com (дата обращения 29.12.2017);
16. Inno Setup Installer Documentation [Электронный ресурс] // Режим доступа: http://jrsoftware.org/ ishelp/ (дата обращения 29.12.2017);