

Методические указания для выполнения курсовой работы по курсу «Управление качеством программного обеспечения»

1. Цель работы

В процессе реализации курсового проекта обучающийся должен овладеть современными инструментами, процессами и методиками тестирования. В качестве основного набора рассматриваются Github (управление исходным кодом и управление ошибками), Travis (CI/CD) и PMBOK (в части управления качеством).

2. Введение

Система контроля версий или VCS может значительно облегчить работу разработчиков, пытающихся проанализировать изменения и вклады в общий код. Таким образом, система контроля версий – это ключевой элемент в системе управления настройками программного обеспечения, которые отвечают потребностям проекта. VCS дают возможность назначать для определенных изменений/ревизий/обновлений буквенные или числовые значения. Также могут предоставить информацию о временных метках и идентификаторе человека внесшего изменения. Самой часто используемой VCS является GIT. Для размещения GIT репозитория в сети, необходимо использовать веб-хостинги, крупнейшим из них является GITHUB.


GIT работает с репозиториями – набором файлов.


3. Регистрация в Github


- Перейти на сайт <https://github.com/>.
- Создать аккаунт (Рисунок 1).
- Выбрать вид подписки **Unlimited public repositories for free.**
- Заполнить анкету про цели использования аккаунта или пропустить этот шаг.

Join GitHub

The best way to design, build, and ship software.

**Step 1:**
Create personal account

**Step 2:**
Choose your plan

**Step 3:**
Tailor your experience

Create your personal account

Username

Test211112

✓

This will be your username. You can add the name of your organization later.

Email address

test@t2est.com

✓

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password

.....

✓

Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

Create an account

You'll love GitHub

Unlimited collaborators
Unlimited public repositories

✓

 Great communication

✓

 Frictionless development

✓

 Open source community

Рисунок 1 - Регистрация в github

4. Создание репозитория


- Перейти на <https://github.com/new>.
- Заполнить имя репозитория и нажать на кнопку создания (Рисунок 2).

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name


 Jukerony ▾

 /


Test ✓

Great repository names are short and memorable. Need inspiration? How about **bug-free-invention**.

Description (optional)

☒  Public

Anyone can see this repository. You choose who can commit.


☐  Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾ 

Create repository

Рисунок 2 - Создание репозитория

5. Установка и конфигурация Git

- Перейти на <https://git-scm.com/download/win>, скачиваем git под свою версию ОС.
- Устанавливаем.
- Перезагружаем компьютер.
- Открываем командную строку windows (win+R -> cmd)
- Переходим в папку в которой планируется инициализировать git репозиторий (cd “путь к проекту который планируется выложить в git”).
- Выполняем команду **git init** для инициализации git.
- Выполнить команду **git add -A** для того чтобы добавить в контроль версий все файлы, лежащие в директории.
- Выполнить команду **git commit -m "first commit"**. Для создания контрольной точки состояния репозитория, к которой потом можно будет вернуться.

- Выполнить команду **git remote add origin** путь_к_проекту. Путь к репозиторию имеет формат https://github.com/{логин_гитхаб}/{имя_репозитория}.
- Выполнить команду **git push -u origin master**. Для отправки изменения на github. При выполнении данной команды git запросит логин и пароль к github, которые необходимо ввести.
- Детальнее с git можно ознакомиться по ссылке <https://git-scm.com/book/ru/v2>

6. Создание ISSUES

Issues представляет собой механизм для контроля выполнения задач по проекту, исправлению ошибок.

Для того чтобы перейти к списку Issues в github необходимо перейти к своему проекту и затем перейти на вкладку Issues (Рисунок 3).

Для создания новой Issue необходимо нажать на кнопку New Issue.

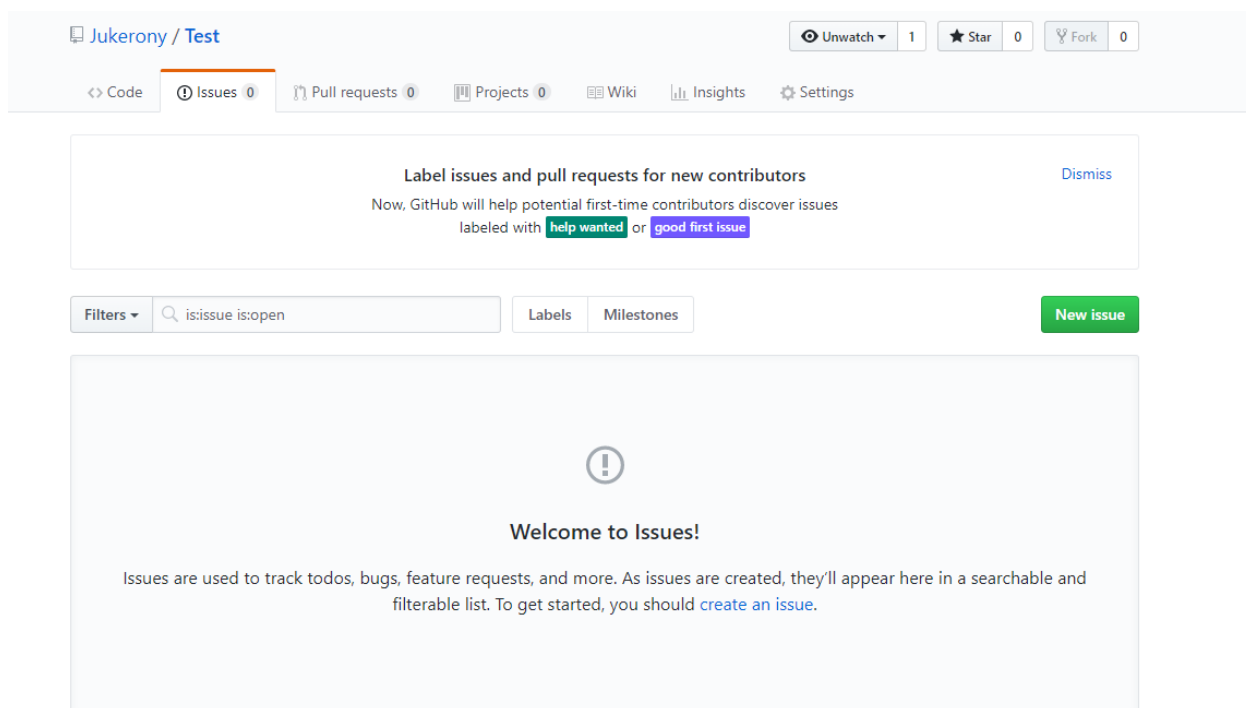


Рисунок 3 - Список Issues

Issue состоит из:

- названия, название должно быть понятным для разработчиков, задействованных в разработке, оно не должно быть слишком развернутым, но при этом должно доносить суть описываемой внутри проблемы;

- описания с использованием языка разметки Markdown (необязательно), описание должно развернуто описывать решаемую проблему;
- назначения на пользователей (необязательно). Поле Assignees, отображает пользователей ответственных за решение проблемы.
- меток (необязательно). Метка должна отражать назначение Issue, например, если это баг, то метка bug, если доработка, то можно сделать соответствующую метку;
- связи с проектами (необязательно). Для организации контроля за исполнения Issues в рамках определенного проекта можно установить Issue на определенный проект. Для этого сначала надо создать проект во вкладке Projects
- майлстоунов. Milestone (веха) — в управлении проектами контрольная точка, значимый, ключевой момент, (например, переход на новую стадию, новый этап в ходе выполнения проекта).

7. Непрерывная интеграция с Travis

Концепция непрерывной интеграции и доставки (CI/CD) — концепция, которая реализуется как конвейер, облегчая слияние только что закомиченного кода в основную кодовую базу. Концепция позволяет запускать различные типы тестов на каждом этапе (выполнение **интеграционного** аспекта) и завершать его запуском с развертыванием закомиченного кода в фактический продукт, который видят конечные пользователи (выполнение **доставки**).

Непрерывная интеграция (англ. Continuous Integration, далее CI) — это практика разработки программного обеспечения, которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения проблем интеграции результатов работы нескольких разработчиков.

Travis CI — распределённый веб-сервис для сборки и тестирования программного обеспечения, использующий GitHub в качестве хостинга исходного кода. Веб-сервис поддерживает сборку проектов на множестве языков, включая C, C++, D, JavaScript, Java, PHP, Python и Ruby. Для начала работы с Travis необходимо:

- используя GitHub аккаунт зайти на <https://travis-ci.org/auth>;
- дать разрешения Travis по доступу к персональным данным;
- включить проект для интеграции в профиле Travis (рисунок 4);
- добавить файл конфигурации (.travis.yml) для языка используемого в проекте, в github репозиторий. Помощь в настройке файла

конфигурации для определенного языка можно найти по ссылке <https://docs.travis-ci.com/user/languages>

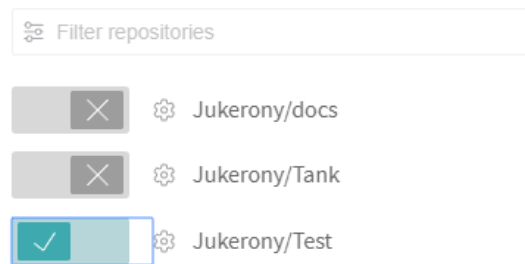


Рисунок 4 - включение интеграции в github репозитории

Теперь при каждом **git push**, Travis будет выполнять сборку проекта. Состояния сборки вы можете оценить по ссылке https://travis-ci.org/{логин}/{имя_репозитория}.

8. Задание на курсовой проект

1. Взять свою лабораторную работу по этому курсу и создать для нее репозиторий. Выложить на github.
2. Разработать план выпуска новой версии (только содержательную часть).
3. Оформить работы по выпуску новой версии в виде списка issues.
4. Создать скелет функционала новой программы (классы и экранные формы).
5. Создать тесты на новый функционал. Отделить тесты на реализованный функционал от нереализованных.
6. Создать план управления качеством (артефакт согласно PMBOK).
7. Настроить систему непрерывной интеграции с помощью Travis, GitHub Actions и иных подобных систем. Сделать так, чтобы тесты на реализованный функционал выполнялись в ответ на интеграцию.
8. Создать отчет по курсовому проектированию.