МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные технологии»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «КОЛЛЕКТИВНАЯ РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ»

Ростов-на-Дону
ДГТУ
2018

УДК 004.4’2, 004.75

Составитель М.В. Ступина

Методические указания для выполнения лабораторных работ по дисциплине «Коллективная разработка информационных систем». – Ростов-на-Дону: Донской гос. техн. ун-т, 2018. – 10 с.

Рассмотрены вопросы использования программного обеспечения и методологий разработки информационных систем командой полипрофессиональных разработчиков.

Содержат сведения о структуре дисциплины, ее содержании, а также рекомендации по изучению дисциплины.

Предназначены для обучающихся направления 09.03.03 Прикладная информатика очной и заочной форм обучения.

УДК 004.4’2, 004.75

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

Ответственный за выпуск зав. кафедрой «Информационные технологии»,
д-р техн. наук, профессор Б.В. Соболь

В печать 26.12. 2018 г.
Формат 60×84/16. Объем 0,6 усл.п.л.
Тираж 50 экз. Заказ № 2016.

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина,1

Лабораторная работа №1
ПЕРВОНАЧАЛЬНАЯ НАСТРОЙКА GIT

Цель работы: провести первоначальную настройку системы контроля вер- сии git, после установки инициализировать каталог для работы, разобраться с су- ществующими состояниями файлов в git, сделать первый коммит.

Форма отчета: демонстрация выполненного задания преподавателю.

Методические указания:

1. Изучить теоретическую часть работы.
2. Выполнить задания.
3. Продемонстрировать пример работы преподавателю.

В состав git'a входит утилита git config , которая позволяет просматривать и устанавливать параметры, контролирующие все аспекты работы git'a и его внешний вид.

Первое, что необходимо сделать после установки git'a, — указать имя и адрес электронной почты. Это важно, потому что каждый коммит в git'e содержит эту информацию, и она включена в коммиты, передаваемые разработчиками, и не может быть далее изменена.

Для того чтобы начать использовать git для существующего проекта, необхо- димо перейти в проектный каталог и в командной строке ввести git init.

Эта команда создаёт в текущем каталоге новый подкаталог с именем .git со- держащий все необходимые файлы репозитория — основу git-репозитория. На этом этапе проект ещё не находится под версионным контролем. Данная команда ини- циализирует возможность работы с git, но не вносит файлы под контроль.

Задания для работы:

1. Зайти в папку T://{Номер группы} и в ней создать папку соответствующую инициалам студента на английском языке. Например, для студента Иванов Петр Петрович, папка будет иметь имя IPP.
2. Провести инициализацию репозитория в созданной папке. Для этого, открыть программу Git Bash, перейти в созданную папку (для перемещения используется команда cd T://{Номер группы}/{Инициалы}).
3. Установить настройки имени и e-mail’a, не используя опцию --global.
4. Создать в папке файл my_first_file.txt и проиндексировать его.
5. Сделать первый коммит.
6. Открыть файл my_first_file.txt и добавить в него строчку “test row”. Проиндексировать изменения.
7. Создать новый файл my_second_file.txt. Проиндексировать изменения.
8. Сделать второй коммит.

Лабораторная работа №2
ИГНОРИРОВАНИЕ, СРАВНЕНИЕ, УДАЛЕНИЕИ ПЕРЕМЕЩЕНИЕ ФАЙЛОВ

Цель работы: научиться исключать файлы, которые нет необходимости вести в системе контроля версий. Получить практические навыки сравнения проделанных изменений в файлах.

Форма отчета: демонстрация выполненного задания преподавателю.

Методические указания:

1. Изучить теоретическую часть работы.
2. Выполнить задания.
3. Продемонстрировать пример работы преподавателю.

Зачастую, имеется группа файлов, которые не только нет необходимости автоматически добавлять в репозиторий, но и видеть в списках неотслеживаемых. К таким файлам обычно относятся автоматически генерируемые файлы (различные логи, результаты сборки программ и т.п.). В таком случае, необходимо создать файл .gitignore с перечислением шаблонов соответствующих таким файлам.
К шаблонам в файле .gitignore применяются следующие правила:

- Пустые строки, а также строки, начинающиеся с # (символ коммента рия), игнорируются.
- Можно использовать стандартные glob шаблоны.
- Можно заканчивать шаблон символом слэша (/) для указания каталога.
- Можно инвертировать шаблон, использовав восклицательный знак (!) в качестве первого символа.

Задания для работы:

1. Продолжить работу с созданным репозиторием на первой лабораторной работе.
2. Создать папку temp в своем репозитории.
3. Создать папку log и добавить в нее 2 файла: main.html и some.tmp.
4. Создать файл .gitignore и добавить в игнорирование папку temp и файлы с расширением .tmp из папки log.
5. Закоммитить добавление файла .gitignore.
6. Внести изменения в файл my_first_file.txt, добавив строчку “row to index”, проиндексировать данные изменения. Еще раз внести изменения в файл, добавив строчку “row no index”.
7. Посмотреть индексированные и неиндексированные изменения используя команду git diff.
8. Удалить файл my_first_file.txt, зафиксировать данное удаление.

Лабораторная работа №3
ПРОСМОТР ИСТОРИИ КОММИТОВ

Цель работы: освоить механизм работы с командой git log для получения информации об истории коммитов.
Форма отчета: демонстрация выполненного задания преподавателю.

Методические указания:

1. Изучить теоретическую часть работы.
2. Выполнить задания.
3. Продемонстрировать пример работы преподавателю.

По умолчанию, без аргументов, git log выводит список коммитов созданных в данном репозитории в обратном хронологическом порядке. То есть самые последние коммиты показываются первыми.

Один из наиболее полезных параметров — это -p, который показывает дельту (разницу/diff), привнесенную каждым коммитом. Также можно использовать -2, что ограничит вывод до 2-х последних записей.

Задания для работы:

1. Продолжить работу с созданным репозиторием.
2. Изучить возможности команды git log, выполнить различные варианты вывода информации и ее отбора.
3. Выполнить задание согласно варианту.

Номер	Задание
1	Вывести коммиты, автором которых являетесь Вы, за последний месяц.
2	Вывести все коммиты в формате: короткий хеш, автор, комментарий.
3	Вывести все коммиты, в сообщении которых присутствует слово my.
4	Вывести все коммиты за текущий месяц с информацией о том, какие файлы были изменены.
5	Вывести информацию о первом коммите в системе, с выводом дельты (diff).
6	Вывести коммиты сделанные за последний месяц назад, но исключая последнюю неделю.
7	Вывести информацию о коммитах в формате: автор, дата, список измененных файлов.
8	Вывести коммиты, автором которых являетесь Вы, с выводом дельты (diff).
9	Вывести коммиты, в которых происходили изменения файла my_first_file.txt, за последние 2 недели.
10	Вывести последние 3 коммита в формате: автор, комментарий.
11	Вывести информацию о первом коммите в формате: дата, автор, комментарий, а также список измененных файлов.
12	Вывести коммиты, автором которых являетесь Вы, со списком измененных файлов.
13	Вывести все коммиты за текущий месяц в формате: сокращенный хеш, дата, комментарий.
14	Вывести все коммиты в формате: e-mail автора, дата коммита, хеши родительских коммитов.
15	Вывести коммиты, в которых происходили изменения файла my_first_file.txt.

Лабораторная работа №4
ОТМЕНА ИЗМЕНЕНИЙ. РАБОТА С МЕТКАМИ

Цель работы: научиться отменять сделанные изменения, работать с метками.
Форма отчета: демонстрация выполненного задания преподавателю.

Методические указания:

1. Изучить теоретическую часть работы.
2. Выполнить задания.
3. Продемонстрировать пример работы преподавателю.

Одна из типичных отмен происходит тогда, когда коммит сделан слишком рано, например, не были добавлены какие-либо файлы, или перепутан комментарий к коммиту. Если необходимо сделать этот коммит ещё раз, можно выполнить git commit с опцией –amend.

Эта команда берёт индекс и использует его для коммита. Если после последнего коммита не было никаких изменений (например, приведенная команда была запущена сразу после предыдущего коммита), то состояние проекта будет абсолютно таким же и всё, что изменится, это комментарий к коммиту.

Git использует два основных типа меток: легковесные и аннотированные. Легковесная метка — это что-то весьма похожее на ветку, которая не меняется — это просто указатель на определённый коммит. А вот аннотированные метки хранятся в базе данных Git'a как полноценные объекты. Они имеют контрольную сумму, содержат имя поставившего метку, e-mail и дату, имеют комментарий и могут быть подписаны и проверены с помощью GNU Privacy Guard (GPG). Обычно рекомендуется создавать аннотированные метки, чтобы иметь всю перечисленную информацию; но если необходимо сделать временную метку или по какой-то причине нет необходимости сохранять остальную информацию, то для этого годятся и легковесные метки.

Задания для работы:

1. Продолжить работу с созданным репозиторием.
2. Создать три файла: 1.txt, 2.txt, 3.txt.
3. Проиндексировать первый файл и сделать коммит с комментарием “add 1.txt file”.
4. Проиндексировать второй и третий файлы.
5. Удалить из индекса второй файл.
6. Перезаписать уже сделанный коммит с новым комментарием “add 1.txt and 3.txt”
7. Создать аннотированную метку с названием v0.01.
8. Создать легковесную ветку указывающую на первый коммит в репозитории.

Лабораторная работа №5
ВЕТВЛЕНИЕ. КОНФЛИКТЫ

Цель работы: научиться работать с ветками, решение конфликтов.

Форма отчета: демонстрация выполненного задания преподавателю.

Методические указания:

1. Изучить теоретическую часть работы.
2. Выполнить задания.
3. Продемонстрировать пример работы преподавателю.

Ветка в git'e — это просто легковесный подвижный указатель на один из коммитов. Ветка по умолчанию в git'e называется master . Когда происходит создание коммита на начальном этапе, доступна ветка master, указывающая на последний сделанный коммит. При каждом новом коммите она сдвигается вперёд автоматически.

Для того чтобы создать новую ветку используется команда git branch.

Эта команда создаст новый указатель на тот самый коммит, на котором сейчас находится git.

Задания для работы:

1. Продолжить работу с созданным репозиторием.
2. Создать новую ветку my_first_branch.
3. Перейти на ветку и создать новый файл in_branch.txt, закоммитить изменения.
4. Вернуться на ветку master.
5. Создать и сразу перейти на ветку new_branch.
6. Сделать изменения в файле 1.txt, добавить строчку “new row in 1.txt file”, закоммитить изменения.
7. Перейти на ветку master и слить ветки master и my_first_branch, после чего слить ветки master и new_branch.
8. Удалить ветки my first branch и new_branch.
9. Создать ветки branch_1 и branch_2.
10. Перейти на ветку branch_1 и изменить файл 1.txt, удалить все содержимое и добавить текст “fix in 1.txt”, изменить файл 3.txt, удалить все содержимое и добавить текст “fix in 3.txt”, закоммитить изменения.
11. Перейти на ветку branch_2 и также изменить файл 1.txt, удалить все содержимое и добавить текст “My fix in 1.txt”, изменить файл 3.txt, удалить все содержимое и добавить текст “My fix in 3.txt”, закоммитить изменения.
12. Слить изменения ветки branch_2 в ветку branch_1.
13. Решить конфликт файла 1.txt в ручном режиме, а конфликт 3.txt используя команду git mergetool с утилитой Meld.

Лабораторная работа №6
ПРЯТАНЬЕ

Цель работы: изучить механизм прятанья.
Форма отчета: демонстрация выполненного задания преподавателю.

Методические указания:

1. Изучить теоретическую часть работы.
2. Выполнить задания.
3. Продемонстрировать пример работы преподавателю.

Часто возникает такая ситуация, что пока идет работа над частью своего проекта, всё находится в беспорядочном состоянии, а нужно переключить ветки, чтобы немного поработать над чем-то другим. Проблема в том, что делать коммит с наполовину доделанной работой только для того, чтобы позже можно было вернуться в это же состояние не хотелось бы. Ответ на эту проблему — команда git stash.

Прятанье поглощает грязное состояние рабочего каталога, то есть изменённые отслеживаемые файлы и изменения в индексе, и сохраняет их в стек незавершённых изменений, которые потом в любое время можно снова применить.

Задания для работы:

1. Продолжить работу с созданным репозиторием.
2. Проверить какие ветки слиты с веткой master, а какие нет.
3. Удалить все ветки слитые с master.

4. Создать новую ветку work и перейти в нее.
5. Изменить файл 1.txt.
6. Спрятать данные изменения.
7. Развернуть обратно данные изменения с опцией --index.
8. Развернуть спрятанные изменения в новую ветку.

Лабораторная работа №7
РАБОТА С УДАЛЕННЫМ РЕПОЗИТОРИЕМ

Цель работы: изучить механизм работы с удаленным репозиторием GitHub.
Форма отчета: демонстрация выполненного задания преподавателю.

Методические указания:
1. Изучить теоретическую часть работы.
2. Выполнить задания.
3. Продемонстрировать пример работы преподавателю.

Если необходимо получить копию существующего репозитория Git, например, проекта, в котором разработчик планирует поучаствовать, то необходимо ис- пользовать команду git clone. Каждая версия каждого файла из истории проекта забирается (pulled) с сервера, когда выполняется команда git clone. Фактически, если серверный диск выйдет из строя, можно использовать любой из клонов на любом из клиентов, для того чтобы вернуть сервер в то состояние, в котором он находился в момент клонирования.
Клонирование репозитория осуществляется командой git clone [url].

Задания для работы:
1. Продолжить работу с созданным репозиторием.
2. Пройти регистрацию на сайте github.com.
3. Настроить доступ к github по SSH.
4. Склонировать репозиторий git@github.com:mavose/tusur_{номер группы}.git.

Лабораторная работа №8
ГИБКИЕ МЕТОДОЛОГИИ РАЗРАБОТКИ

Цель работы: изучить гибкие методологии разработки программного обеспечения.
Форма отчета: демонстрация выполненного задания преподавателю.

Методические указания:
1. Изучить теоретическую часть работы.
2. Выполнить задания.
3. Продемонстрировать пример работы преподавателю.

Гибкие" (agile) методы разработки *ПО* появились как *альтернатива* формальным и тяжеловесным методологиям, наподобие *CMM* и *RUP*. Основными положениями гибких методов, закрепленных в *Agile Manifesto* в 2007 году являются следующее:

- индивидуалы и взаимодействие вместо процессов и программных средств;

- работающее ПО вместо сложной документации;
- взаимодействие с заказчиком вместо жестких контрактов;
- реакция на изменения вместо следования плану.

Задания для работы:

1. Подготовить доклад и презентацию об одной из гибких методологий разработки (Scrum, Kanba, Lean) группой студентов. Тема согласуется дополнительно с преподавателем.

Лабораторная работа №9
ПЛАНИРОВАНИЕ AGILE-ПРОЕКТА

Цель работы: освоить базовые принципы Agile-подхода к разработке программного обеспечения.

Форма отчета: демонстрация выполненного задания преподавателю.

Методические указания:

1. Изучить теоретическую часть работы.
2. Выполнить задания.
3. Продемонстрировать пример работы преподавателю.

Agile – это гибкая методология разработки (англ. Agile software development). Представляет собой революционную концепцию, в рамках которой выполняется разработка программного обеспечения.

Одним из основных подходов является Scrum.

Над каждым проектом работает универсальная команда специалистов, к которой присоединяется еще два человека: владелец продукта и scrum-мастер. Первый соединяет команду с заказчиком и следит за развитием проекта; это не формальный руководитель команды, а скорее куратор. Второй помогает первому организовать бизнес-процесс: проводит общие собрания, решает бытовые проблемы, мотивирует команду и следит за соблюдением scrum-подхода.

Scrum-подход делит рабочий процесс на равные спринты – обычно это периоды от недели до месяца, в зависимости от проекта и команды. Перед спринтом формулируются задачи на данный спринт, в конце – обсуждаются результаты, а команда начинает новый спринт. Спринты очень удобно сравнивать между собой, что позволяет управлять эффективностью работы.

Задания для работы:

1. Зарегистрироваться на сайте jazz.net – всем студентам.
2. Создать проект – всем или одному из подгруппы.
3. Добавить в проект зарегистрированных ранее участников и назначить им роли (скрам-мастера и участников проекта) – выполняет владелец проекта (администратор).
4. Выполнить планирование проекта и отслеживание процесса его выполнения по ролям

Лабораторная работа №10
КОМАНДНАЯ РАЗРАБОТКА

Цель работы: выполнить разработку информационной системы совместной группой студентов.

Форма отчета: демонстрация выполненного задания преподавателю.

Методические указания:

1. Изучить теоретическую часть работы.
2. Выполнить задания.
3. Продемонстрировать пример работы преподавателю.

Командная работа практически всегда воспринимается как нечто исключительно положительное, ведь если работать сообща, есть возможность достичь синергии и реализовать те проекты и выполнить те задачи, которые для одного человека будут слишком «неподъёмными». И результат в командной работе достигается только тогда, когда усилия всех её членов устремлены в одном направлении. И ответственность за все результаты также несёт группа людей, а не просто один человек.

В первую очередь следует заметить, что командная работа представляет собой одну из форм делегирования полномочий. Совсем необязательно, чтобы в команде присутствовали сотрудники одного статуса – у всех членов команды могут быть свои должности и полномочия, однако обязанности и права всегда должны быть едиными для всех. И это очень важно, ведь, несмотря на различие в статусе, у всех «игроков» должны быть одинаковые позиции.

Что же касается одинаковых обязанностей и прав, то это равенство представляет собой основополагающий принцип командной работы, т.к. посредством него можно дать объективную характеристику потенциала всех членов команды в рассматриваемой форме деятельности.

И чтобы команда была успешной, а её функционирование давало соответствующие результаты, необходимо, чтобы соблюдались нижеследующие условия:

- Грамотная постановка целей
- Наличие чётких и ясных задач
- Правильный подбор командного состава
- Наличие детализированной системы работы
- Способность людей работать в команде

Задания для работы:

На основе спланированного проекта с использованием удаленного репозитория выполнить разработку программного продукта.

ЛИТЕРАТУРА

1. https://www.intuit.ru/studies/courses/3490/732/info
2. https://www.intuit.ru/studies/courses/582/438/info
3. https://www.intuit.ru/studies/courses/4806/1054/info