

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**(ДГТУ)**

|  |  |
| --- | --- |
| Факультет | Информационно-экономических систем |
|  | (наименование факультета) |
| Кафедра | Информационные системы в строительстве |
|  | (наименование кафедры) |

**ОТЧЕТ**

по учебной практике: практика по получению первичных профессиональных умений и навыков

на тему:

изучение основ локальной работы с VCS GIT

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Автор |  | | |  | Боровкова М.С | | | |
|  | (подпись, дата) | | |  |  | | | |
| Обозначение | 09.03.01.930000.000 О | | Группа | | | | ВИп11 |
| Направление подготовки | | 09.03.01 Информатика и вычислительная техника | | | | | |
| Профиль | Информатика и вычислительная техника | | | | | | |
| Руководитель |  | | |  | | О.В. Шиляева | |
|  | (подпись, дата) | | |  | |  | |

г. Ростов-на-Дону

2018 год

Содержание

[ВВЕДЕНИЕ 3](#_Toc507842800)

[Глава 1 Особенности хранения данных в Git 5](#_Toc507842801)

[1.1 Основы Git 5](#_Toc507842802)

[1.2 Хранение данных: объекты 6](#_Toc507842803)

[Глава 2 Локальность операций в Git 7](#_Toc507842804)

[2.1 Git – изнутри 7](#_Toc507842805)

[Глава 3 Работа с репозиторием 8](#_Toc507842806)

[Глава 4 Состояние файлов в Git? 14](#_Toc507842807)

[4.1 Виды состояний файлов в Git 14](#_Toc507842808)

[ЗАКЛЮЧЕНИЕ 16](#_Toc507842809)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 17](#_Toc507842810)

# ВВЕДЕНИЕ

Подобно множеству других замечательных программных продуктов, система Git начиналась с небольшого созидательного разрушения и пылкой полемики. Ядро Linux представляет собой крайне масштабный проект ПО с открытым исходным кодом. В истории поддержки ядра Linux изменения программ долгое время передавались в виде исправлений (patches) и архивированных файлов. В 2002 году для проекта Linux стали использовать собственную систему DVCS, которая называлась BitKeeper. В 2005 году отношения между сообществом, разрабатывавшим ядро Linux, и коммерческой фирмой, создавшей BitKeeper, были разорваны и бесплатное использование этой системы контроля версий стало невозможным, что побудило сообщество разработчиков Linux (и в частности создателя этой операционной системы Линуса Торвальдса) начать работу над собственным инструментом, взяв за основу некоторые идеи BitKeeper. Вот цели, которые ставились для новой системы:

 быстродействие;

 простое проектное решение;

 мощная поддержка нелинейной разработки (тысячи параллельных ветвей);

 полностью распределенная система;

 возможность эффективной (в плане быстродействия и объема данных) работы с большими проектами, такими как ядро Linux.

С момента своего появления в 2005 году система Git развивалась и совершенствовалась в попытках добиться простоты использования при сохранении изначальных характеристик. Она работает необыкновенно быстро, крайне эффективна для больших проектов и обладает потрясающей ветвящейся системой нелинейной разработки Система управления версиями (Version Control System) представляют собой программное обеспечение для облегчения деятельности с быстро меняющейся информацией. Система контроля версий предоставляет возможность хранить несколько вариантов одного и того же документа. При необходимости можно вернуться к старым версиям, можно узнать, кем были сделаны те или иные изменения и т.д. Такого рода системы в большинстве своем используются при разработке программного обеспечения, чтобы можно было хранить исходные коды разрабатываемых программ. А также данные системы могут использоваться и в других областях знаний, которые включают в себя огромное количество часто изменяющихся электронных документов.

Целью данного отчета является рассмотрение версии VCS GIT.

# Глава 1 Особенности хранения данных в Git

## 1.1 Основы Git

Главным отличием Git от любой другой системы контроля версий (в том числе Subversion и ей подобных) является восприятие данных. Большинство систем хранит информацию в виде списка изменений, связанных с файлами. Эти системы (CVS, Subversion, Perforce, Bazaar) рассматривают хранимые данные как набор файлов и изменений, которые вносились в эти файлы в течение их жизни.

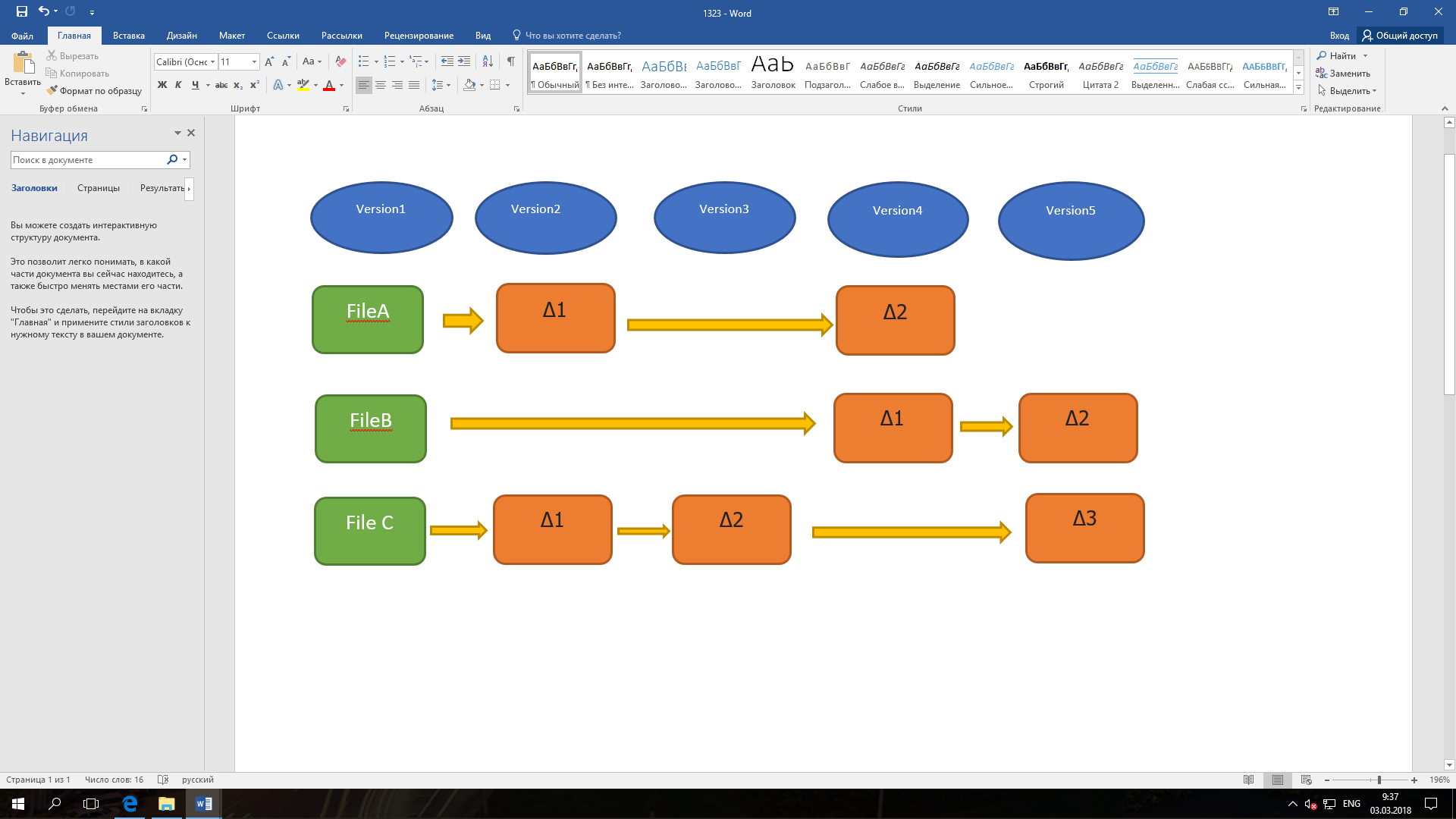


Рисунок – VCS, хранящие изменения файлов

Вместо этого Git всегда хранит всё состояние проекта в виде фиксаций (commits, разг. комитов), которые представляют собой снимок (snapshot) директории именно в тот момент, когда разработчик решил внести изменение. Стоит отметить, что момент внесения изменений и их фиксации, конечно, отличается. Git настолько мощный, что позволяет фиксировать только части внесённых изменений. Тут можно заметить одну сложность: если Git каждый раз создаёт фиксацию всех файлов целиком, то как он хранит неизменённые при фиксации файлы? Решение этого вопроса было простым и гениальным: Git хранит ссылку на ранее фиксированный файл. Это ведёт к полному пересмотрению обычного процесса разработки и даёт Git его популярную возможность легко работать с нелинейными проектами, развитие отдельных частей котоых идёт параллельно.

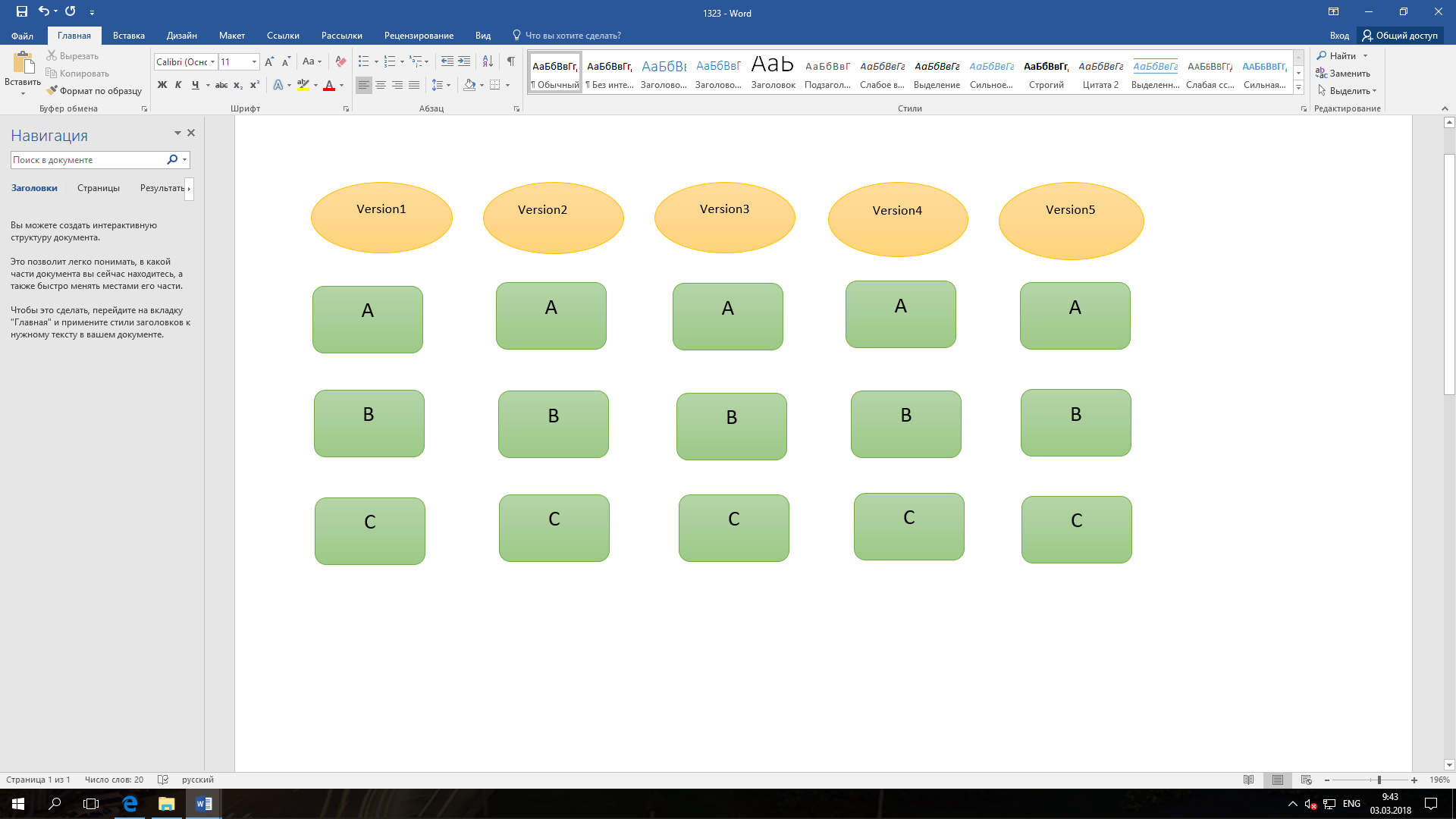


Рисунок 2 – Устройство Git

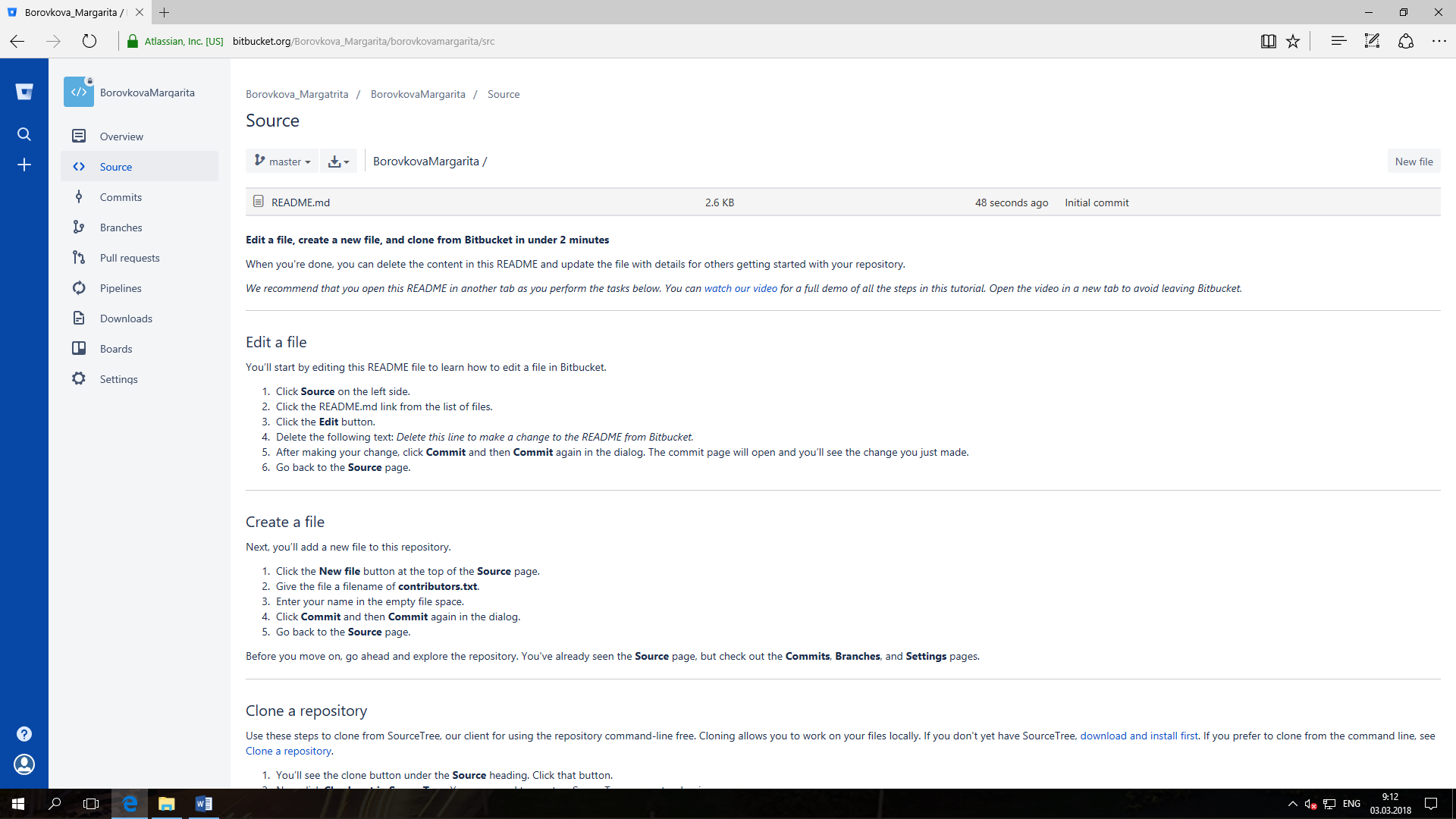


Рисунок 3 – Регистрация на bitbucket.org

## 1.2 Хранение данных: объекты

В Git единицей хранения данных является объект (англ. object), который однозначно определяется 40-символьным хешем sha1. В объектах Git хранит почти всё: коммиты, содержимое файлов, их иерархию. Сначала объекты представляют из себя обычные файлы в папке .git/objects, а после git gc упаковываются в .pack-файлы, о которых будет рассказано чуть ниже. Для экономии дискового пространства содержимое всех объектов дополнительно сжимается с помощью zlib.

# Глава 2 Локальность операций в Git

## 2.1 Git – изнутри

Для осуществления практически всех операций системе Git требуются только локальные файлы и ресурсы – в общем случае информация с других компьютеров сети не нужна. Если вы привыкли к CVCS, где для большинства операций характерны задержки из-за передачи данных по сети, этот аспект работы Git наводит на мысль, что боги скорости наделили эту систему нереальной силой. Когда вся история проекта хранится на локальном диске, кажется, что большинство операций выполняется почти мгновенно. Например, для просмотра истории проекта системе Git нет нужды обращаться к серверу, получать там историю и выводить ее на экран – система просто читает все непосредственно из локальной базы данных. То есть вы видите историю проекта практически сразу же. Если вы хотите посмотреть, чем текущая версия файла отличается от версии месячной давности, Git ищет старый файл и вычисляет внесенные в него правки, вместо того чтобы просить об этой операции удаленный сервер или считывать с этого сервера старую версию файла для локального сравнения. Это также означает, что практически все операции могут проводиться в автономном режиме и без использования виртуальной частной сети (Virtual Private Network, VPN). Если желание поработать появится в самолете или в поезде, вы можете успешно фиксировать изменения, пока не найдете подключение к сети для их выгрузки. Если дома вы не можете добиться корректной работы VPN-клиента, вам это не помешает. Во многих других системах подобное либо невозможно, либо достигается сложным путем и требует больших усилий.

# Глава 3 Работа с репозиторием

1. Подготовка.

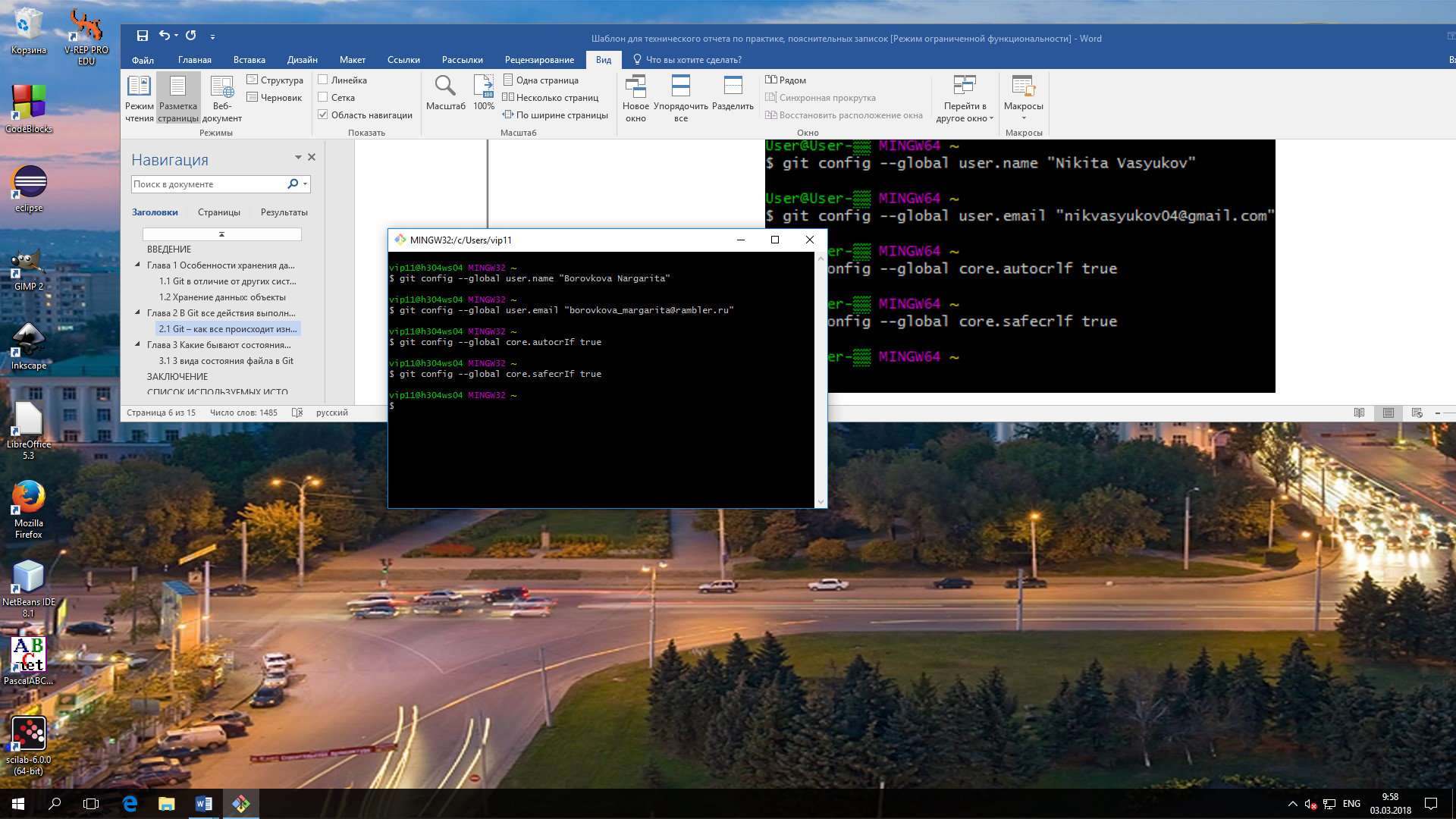


Рисунок 2 – Установка имени, электронной почты

2. Создание проекта.

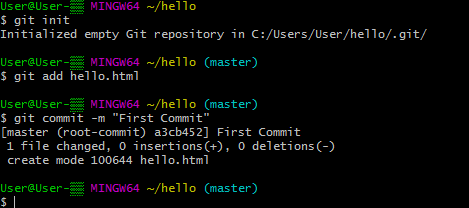


Рисунок 3 – Создание директории, создание страницы «Hello, World!»

3. Проверка состояния.

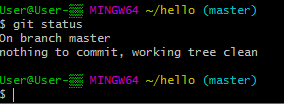


Рисунок 4 – Проверка состояния репозитория

4. Внесение изменений.

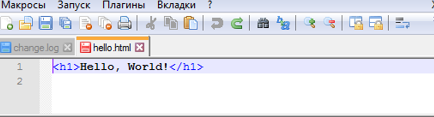


Рисунок 5 – Изменение страницы «Hello, World!»

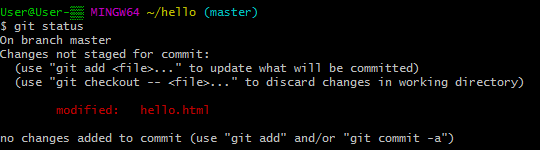


Рисунок 6 – Проверка состояния

5. Индексация изменений.

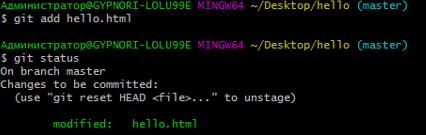


Рисунок 7 – Добавление изменений

6. Коммит изменений.

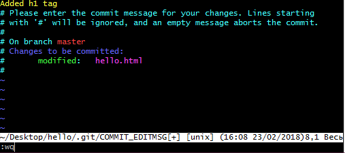


Рисунок 8 – Создание коммита



Рисунок 9 – Проверка состояния коммита



Рисунок 10 – Проверка состояния рабочего каталога

7. Git работает с изменениями, а не с файлами.

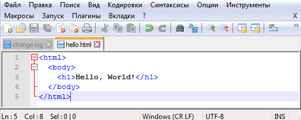


Рисунок 11 – Внесение первого изменения

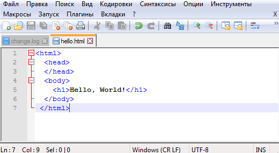


Рисунок 12 – Внесение второго изменения

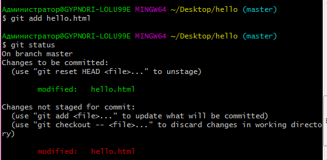


Рисунок 13 – Индексация первого изменения и последующая проверка состояния без индексации второго изменения

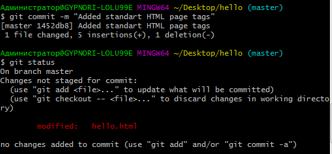


Рисунок 14 – Коммит первого изменения и проверка состояния

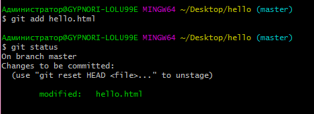


Рисунок 15 – Индексация второго изменения



Рисунок 16 – Коммит второго изменения

8. История.

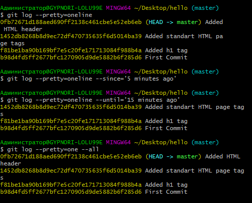


Рисунок 17 – Просмотр истории проекта



Рисунок 18 – Различные форматы лога

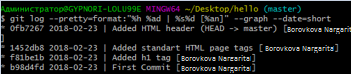


Рисунок – Конечный формат лога

9. Алиасы.

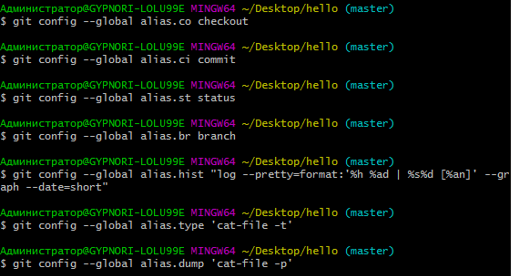


Рисунок 20 – Общие алиасы

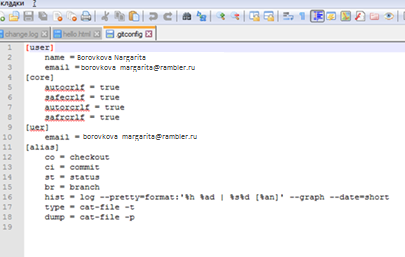


Рисунок 21 – Задание алиаса hist в файле .gitconfig

10. Получение старых версий.

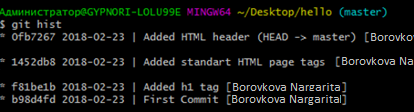


Рисунок 22 – Получение хэша для первого коммита

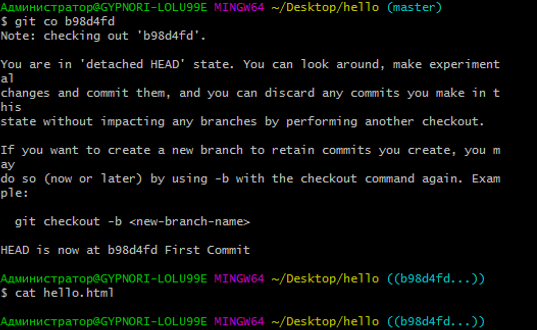


Рисунок 23 – Получение старой версии

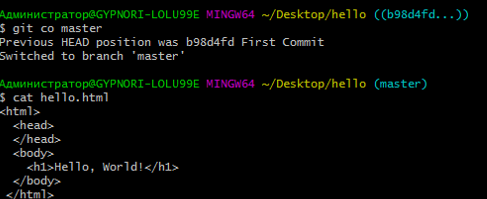


Рисунок 24 – Возвращение к последней версии в ветке master

# Глава 4 Состояние файлов в Git?

## 4.1 Виды состояний файлов в Git

В Git файлы могут находиться в одном из трёх состояний: зафиксированном, изменённом и подготовленном. «Зафиксированный» значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы – это изменённые файлы, отмеченные для включения в следующий коммит.

Таким образом, в проекте с использованием Git есть три части: каталог Git (Git directory), рабочий каталог (working directory) и область подготовленных файлов (staging area).

Каталог Git – это место, где Git хранит метаданные и базу данных объектов вашего проекта. Это наиболее важная часть Git.

Рабочий каталог – это извлечённая из базы копия определённой версии проекта. Эти файлы достаются из сжатой базы данных в каталоге Git и помещаются на диск для того, чтобы вы их просматривали и редактировали.

Область подготовленных файлов – это обычный файл, обычно хранящийся в каталоге Git, который содержит информацию о том, что должно войти в следующий коммит. Иногда его называют индексом (index), но в последнее время становится стандартом называть его областью подготовленных файлов (staging area).

Стандартный рабочий процесс с использованием Git выглядит примерно так:

1. Вы изменяете файлы в вашем рабочем каталоге.

2. Вы подготавливаете файлы, добавляя их слепки в область подготовленных файлов.

3. Вы делаете коммит. При этом слепки из области подготовленных файлов сохраняются в каталог Git.

Если рабочая версия файла совпадает с версией в каталоге Git, файл считается зафиксированным. Если файл изменён, но добавлен в область подготовленных данных, он подготовлен. Если же файл изменился после выгрузки из БД, но не был подготовлен, то он считается изменённым.

# ЗАКЛЮЧЕНИЕ

В работе была рассмотрена система контроля версий Git и использование ее.

Такого рода система позволяет облегчить проекты по разработке программного обеспечения, и в определенной степени исключить человеческие ошибки. Существующие системы мониторинга помогут в случае возникновения каких-либо нестыковок в работе. В настоящее время существует большое количество систем контроля версий, и наиболее интересные и информативные вариант это Git. Пока не существует идеальных систем, каждая из них обладает определенными недостатками, но одновременно с этим и рядом особенностей, которые выделяют ее. На смену относительно старым системам приходят улучшенные альтернативы. Другие системы более приятны и предоставляют лучший рабочий опыт. У потребителя, безусловно, есть много возможностей для выбора. Большинство производителей предлагают демонстрационные версии своих программных продуктов.

Системы контроля версий являются потрясающим решение проблем распространенных корпораций. Безусловно данная область будет развиваться и в дальнейшем возможно увеличение числа систем, и существенная доработка существующих.

В рамках прохождения учебной практики на базе ДГТУ были освоены основные этапы работы с VCS GIT.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. URL: https://nsmubase.ru/bez-rubriki/kursovaya-rabota-sistemy-kontrolya-versij/ (дата обращения: 20.02.2018)

2. URL: https://ru.wikipedia.org/wiki/Git (дата обращения: 20.02.2018)

3. URL: https://habrahabr.ru/company/badoo/blog/163853/ (дата обращения: 20.02.2018)

4. URL: <https://githowto.com/> (дата обращения 20.02.2018)

5. URL: https://git-scm.com/book/ru/v2 (дата обращения: 20.02.2018)

6. URL: <https://bitbucket.org/> (дата обращения 20.02.2018)

7. URL: http://freesource.info/wiki/RuslanHihin/gitusermanual?v=b7s& (дата обращения: 20.02.2018)

8. Чакон С., Штрауб Б. Git для профессионального программиста / С. Чакон, Б. Штрауб – Питер: 2016. – 496 с.