

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины «Основы программной инженерии»

Выполнил:
Плугатырев Владислав Алексеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Доцент кафедры инфокоммуникаций
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Исследование основных возможностей Git и GitHub.

Цель работы: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

Порядок выполнения работы

1. Создал репозиторий.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * **Repository name ***

BigLofanbl4 / Lab-1

✓ Lab-1 is available.

Great repository names are short and memorable. Need inspiration? How about [fuzzy-goggles](#) ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **Node**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **MIT License**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

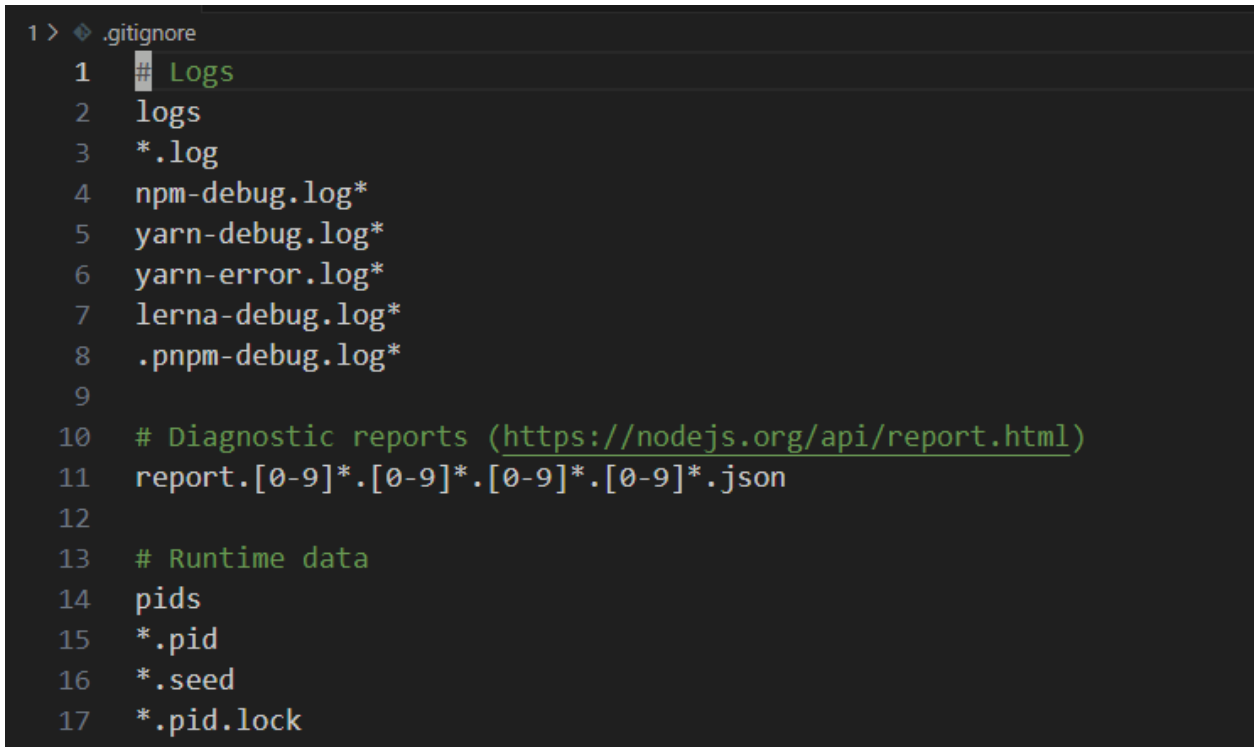
Рисунок 1 - Создание репозитория

2. Клонировал репозиторий на рабочий компьютер.

```
C:\Users\MSI\Desktop\Основы программной инженерии\1>git clone https://github.com/BigLofanbl4/Lab-1.git
Cloning into 'Lab-1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 2 - Клонирование репозитория

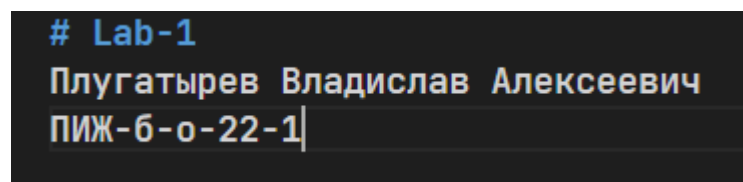
3. Добавил необходимые правила в .gitignore.



```
1 > .gitignore
1  # Logs
2  logs
3  *.log
4  npm-debug.log*
5  yarn-debug.log*
6  yarn-error.log*
7  lerna-debug.log*
8  .pnpm-debug.log*
9
10 # Diagnostic reports (https://nodejs.org/api/report.html)
11 report.[0-9]*.[0-9]*.[0-9]*.[0-9]*.json
12
13 # Runtime data
14 pids
15 *.pid
16 *.seed
17 *.pid.lock
```

Рисунок - 3 Правила в .gitignore

4. Добавил требуемую информацию в README.md.



```
# Lab-1
Плугатырев Владислав Алексеевич
ПИЖ-6-о-22-1|
```

Рисунок - 4 Файл README.md

5. Написал программу. При написании программы фиксировал изменения в локальном репозитории с помощью команды `git commit`.

```
function blackJack() {
  const dillerCards = [getCard(), getCard()];
  const playerCards = [getCard(), getCard()];

  let dillerPoints = sumPoints(dillerCards);
  let playerPoints = sumPoints(playerCards);

  console.log(`Diller has got ${dillerCards[0]}`);
  console.log(`You've got ${playerCards.toString()} (${playerPoints})`);

  while (prompt("Do you want to hit (y/n)?", "y") != "n") {
    playerCards.push(getCard());

    playerPoints = sumPoints(playerCards, playerPoints);

    console.log(`You've got ${playerCards.toString()} (${playerPoints})`);

    if (checkLose(playerPoints)) {
      console.log("You've lost!");
      break;
    }
  }

  dillerPoints = dillerPlay(dillerPoints, playerPoints, dillerCards);
  console.log(`Diller has got ${dillerCards.toString()}, which equals ${dillerPoints}`);

  if (dillerPoints > playerPoints && dillerPoints <= 21) {
    console.log("You've lost");
  } else {
    console.log("You've won");
  }
}

blackJack();
```

Рисунок 5 – Программа

Ответы на контрольные вопросы

1. Системы контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени. В свете усложнения сред разработки они помогают командам разработчиков работать быстрее и эффективнее.

2. Недостатки локальных СКВ: возможность потери данных вследствие возникновения физических поломок оборудования; отсутствие возможности совместной разработки. Недостатки

централизованных СКВ: отсутствие доступа к данным при сбое работы сервера; довольно низкая скорость работы (из-за возникновения сетевых задержек).

3. Распределенные СКВ.

4. Рабочий код хранится на нескольких компьютерах, а история всех версий хранится как на удалённом сервере, так и на каждом из этих компьютеров.

5. В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом.

6. У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged). Если определённая версия файла есть в Git-директории, эта версия считается зафиксированной. Если версия файла изменена и добавлена в индекс, значит, она подготовлена. И если файл был изменён с момента последнего распаковывания из репозитория, но не был добавлен в индекс, он считается изменённым.

7. На странице профиля отображаются сведения о вашей работе через репозитории, которые вас интересуют, вклад, который вы сделали, и беседы, в которых вы участвовали.

8. Публичный (public) и закрытый (private).

9. Регистрация и настройка, создания репозитория, клонирование репозитория, создание и изменение файлов, добавление и фиксация изменений, отправка изменений на GitHub.

10. Первое, что нужно сделать - указать имя и адрес электронной почты пользователя.

11. Нажать на кнопку создания репозитория, дать имя репозиторию, выбрать видимость репозитория, если необходимо отметить флажки создания файлов README и .gitignore.

12. MIT License, GNU General Public License (GPL), Apache License, Creative Commons.

13. Для клонирования необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования. Далее необходимо открыть командную строку или терминал и перейти в каталог, куда необходимо скопировать хранилище. Затем написать `git clone` и ввести адрес. Клонирование репозитория GitHub позволяет сохранить локальную копию проекта, работать с кодом, получать изменения и сотрудничать с другими разработчиками.

14. Использовать команду `git status`.

15. После выполнения следующих операций состояние локального репозитория Git изменяется следующим образом:

- Добавление/изменение файла: Когда добавляется или изменяется файл в локальной репозитории Git, Git определяет этот файл как измененный. Однако он еще не отслеживается и не готов для коммита.

- Добавление файла под версионный контроль: Выполнение команды ``git add`` добавляет указанный файл в индекс Git. Это означает, что файл теперь отслеживается Git и готов для фиксации. Состояние изменения файла изменяется с "изменен" на "отслеживается".

- Фиксация изменений (коммит): Когда выполняется команда ``git commit``, Git создает коммит, содержащий фиксацию изменений. Все файлы, находящиеся в индексе, включаются в коммит. После коммита состояние файлов изменяется на зафиксированный (committed).

- Отправка изменений на сервер: Команда `git push` используется для отправки локальных коммитов на удаленный сервер,

например, на GitHub. После успешного выполнения команды `Git` синхронизирует внешний репозиторий с локальным репозиторием. После отправки изменений состояние удаленного репозитория становится идентичным состоянию локального репозитория.

16. Сначала на первом и втором компьютере необходимо выполнить команду `git clone`. После этого на обоих компьютерах есть локальные копии репозитория, при этом потребуется регулярно синхронизировать изменения между компьютерами и репозиторием GitHub. Для этого необходимо добавить все изменения в локальном репозитории с помощью команды `git add`, затем ввести команду `git commit` для фиксации изменений, после для отправки изменений на GitHub использовать команду `git push`. Чтобы загрузить последние изменения необходимо использовать `git pull`.

17. GitLab, CodeBase, SourceForge. GitHub делает упор на высокую доступность и производительность своей инфраструктуры и делегирует другие сложные функции сторонним инструментам. GitLab, наоборот, фокусируется на включении всех функций на одной проверенной и хорошо интегрированной платформе; он обеспечивает все для полного жизненного цикла DevOps под одной крышей. Что касается популярности, GitHub определенно превосходит GitLab.

18. GitHub Desktop обеспечивает удобный способ работы с Git и Github с помощью графического интерфейса. Можно создавать новые репозитории, клонировать существующие, отслеживать изменения, создавать ветки, коммиты и многое другое. Операции Git, такие как создание веток и коммитов, отображаются в интерфейсе приложения, что упрощает их выполнение и позволяет работать с Git без использования командной строки.