

# ШПАРГАЛКА

## по использованию системы контроля версий git и хранилища репозитория git.csc.sibsutis.ru

1. Что такое git?.....	1
2. Как войти в хранилище репозитория проектов (git.csc.sibsutis.ru)? .....	1
3. Как создать репозиторий для хранения исходного кода приложения в системе git.csc.sibsutis.ru? .....	2
4. Как добавить преподавателя в члены группы разработчиков? .....	4
5. Как создать локальный репозиторий В ПУСТОМ КАТАЛОГЕ? .....	6
6. Как создать репозиторий в каталоге с исходным кодом приложения? .....	7
7. Как создать начальный коммит в основной ветке репозитория? .....	7
8. Как добавить файлы в локальный репозиторий? .....	8
9. Как посмотреть текущее состояние локального репозитория? .....	8
10. Как зафиксировать («закоммитить») информацию о добавленных файлах (их изменении)? .....	9
11. Как отправить коммиты из локального репозитория в хранилище git.csc.sibsutis.ru? .....	10
12. Как создать новую ветку для выполнения практического задания? .....	10

### 1. ЧТО ТАКОЕ GIT?

**Git** — это система хранения файлов и истории их изменения. Обычно используется для сохранения истории работы с исходным кодом проекта, написанном на каком-либо (или каких-либо) языке(ах) программирования, а также для обмена изменениями в этом исходном коде между участниками команды разработчиков.

Командная работа с использованием **git** позволяет каждому разработчику иметь свой собственный репозиторий и обмениваться своими изменениями с коллегами по мере их готовности. Для обмена изменениями создаются специальных хранилища, где сохраняется «мастер»-копия репозитория. На кафедре вычислительных систем таким хранилищем является GitLab. Он доступен по ссылке <https://git.csc.sibsutis.ru>.

### 2. КАК ВОЙТИ В ХРАНИЛИЩЕ РЕПОЗИТОРИЕВ ПРОЕКТОВ (GIT.CSC.SIBSUTIS.RU)?

В адресной строке браузера вводим адрес — <https://git.csc.sibsutis.ru> и попадаем на страницу входа в хранилище (см. Рисунок 1).

Проверяем, что авторизация идет с использованием FreeIPA, вводим логин (FreeIPA Username) и пароль (Password), которые Вы установили на первом занятии, и нажимаем “Sign In”. Если хотите, чтобы на этом рабочем месте Ваши данные были записаны браузером, и при следующем входе в хранилище автоматически осуществлялась регистрация пользователя, то можно поставить галочку «Remember me». Браузер дополнительно может сохранять введенные Вами логин и пароль (обычно он спрашивает делать это или нет) и, если Вы сохранили эти параметры, то в следующий раз эти данные при регистрации в системе будут автоматически подставляться в поля для ввода.

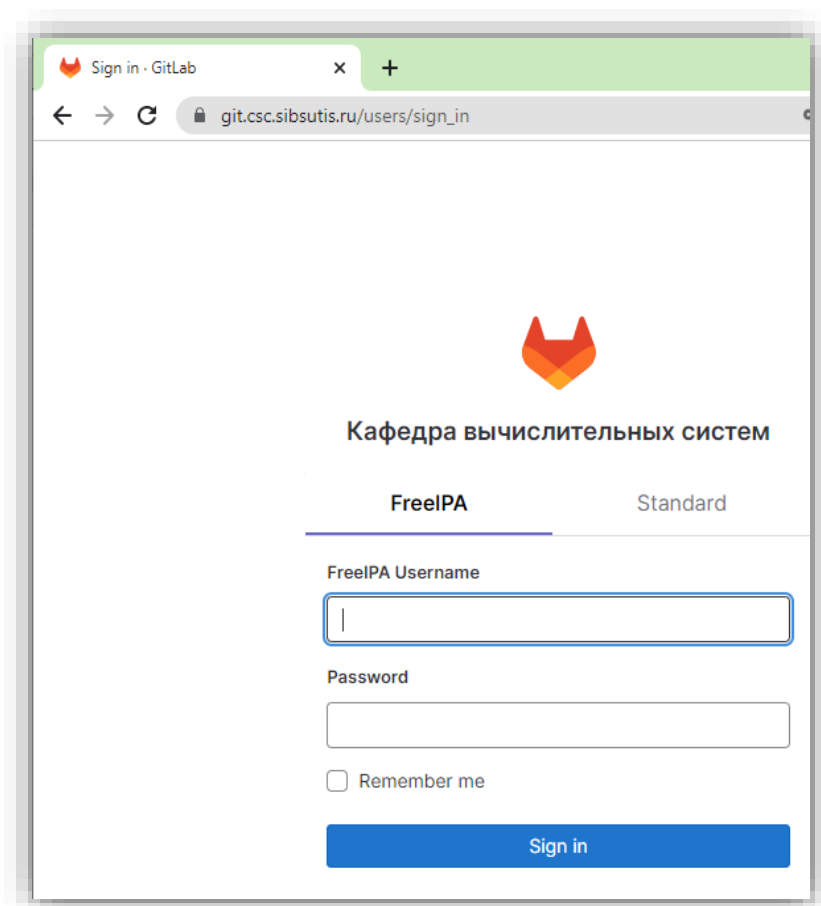


Рисунок 1 – Страница регистрации в системе хранения репозитория git.csc.sibsutis.ru.

### 3. КАК СОЗДАТЬ РЕПОЗИТОРИЙ ДЛЯ ХРАНЕНИЯ ИСХОДНОГО КОДА ПРИЛОЖЕНИЯ В СИСТЕМЕ GIT.CSC.SIBSUTIS.RU?

При первом входе в систему хранения репозитория откроется страница (см. Рисунок 2), на которой одним из предлагаемых дальнейших действий будет «Create a project». В остальных случаях для создания проекта необходимо выбрать в меню значок «плюс» и в выпадающем меню выбрать «New project / repository».

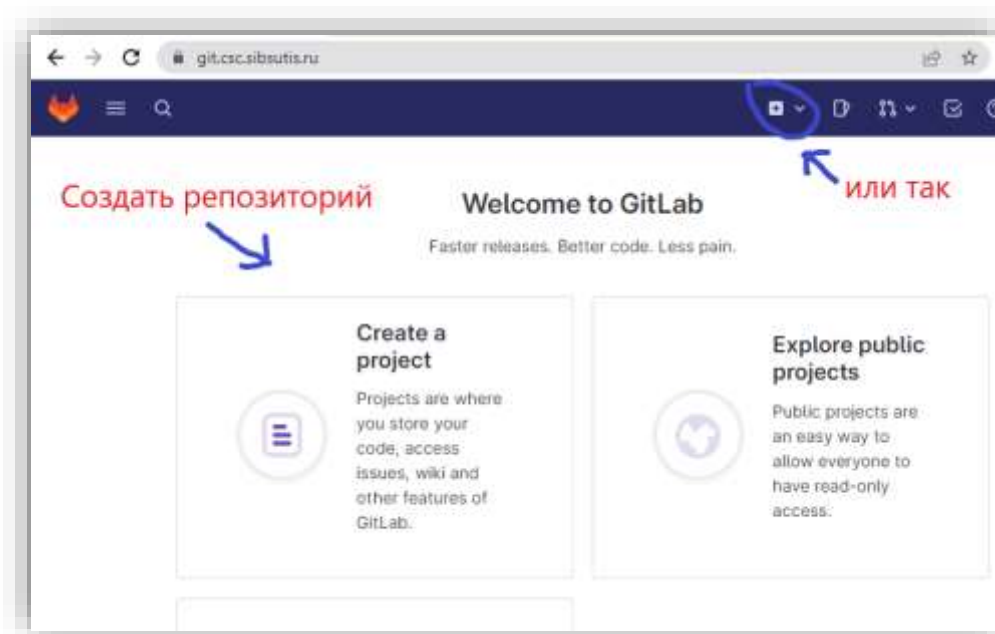


Рисунок 2 – Страница, отображаемая после первого входа в систему хранения репозитория

Начав создавать репозиторий, система предложит Вам выбрать один из вариантов создания репозитория (см. Рисунок 3):

- «Создать пустой репозиторий (Create a project)». Этот вариант используется когда команда разработчиков впервые создает общий репозиторий и репозиторий изначально должен быть пустой. Данные в репозиторий размещает команда самостоятельно;
- «Создать репозиторий по шаблону (Create from template)». Этот вариант предполагает, что репозиторий тоже создается впервые, но команда разработчиков хочет его заполнить сразу каким-то типовым содержимым. Дальнейшее изменение репозитория уже осуществляется командой самостоятельно.
- «Импортировать уже готовый проект (Import project)». Этот пункт используется если требуется перенести проект из другой системы хранения репозитория.

Выбрав «Создать пустой репозиторий» вы попадете на страницу с первоначальными настройками репозитория (см. Рисунок 4). На этой странице необходимо задать имя репозитория (Project name), путь к репозиторию (Project URL), выбрать способ доступа к репозиторию (Visibility Level) и задать начальные настройки, которые может система сделать для репозитория (Project Configuration).

Имя репозитория — это символьная последовательность, которая будет идентифицировать Ваш репозиторий. Оно может состоять из английских букв разного регистра (a-z или A-Z), цифр (0-9), символов-эмодзи (😞, 😊, 🚩 и т.п.), нижнее подчеркивание (\_), тире (-), точки (.), пробела ( ). Имя обязательно должно начинаться с буквы, цифры, символа эмоции или символа подчеркивания. У Вас не может быть двух репозитория с одинаковыми именами.

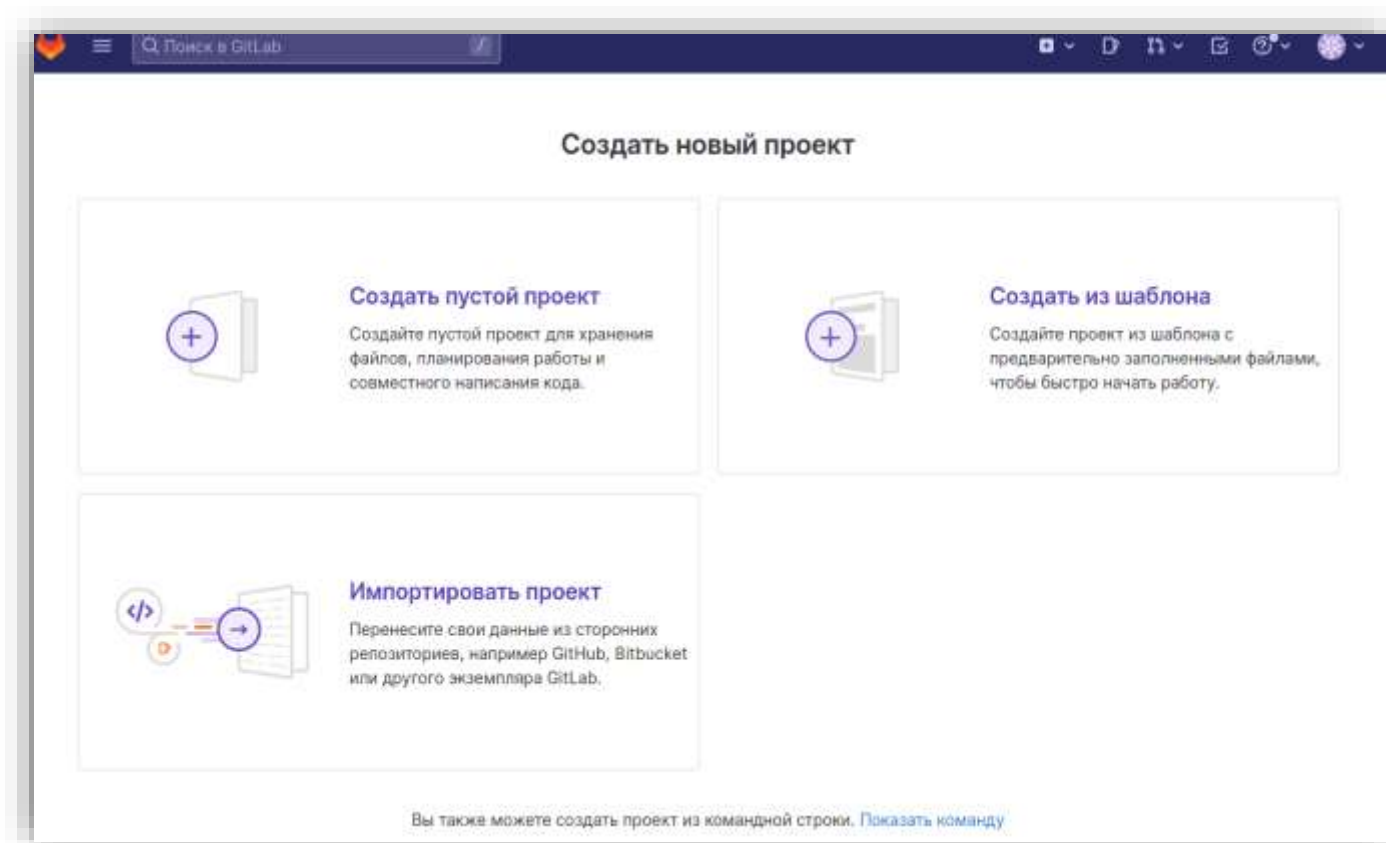
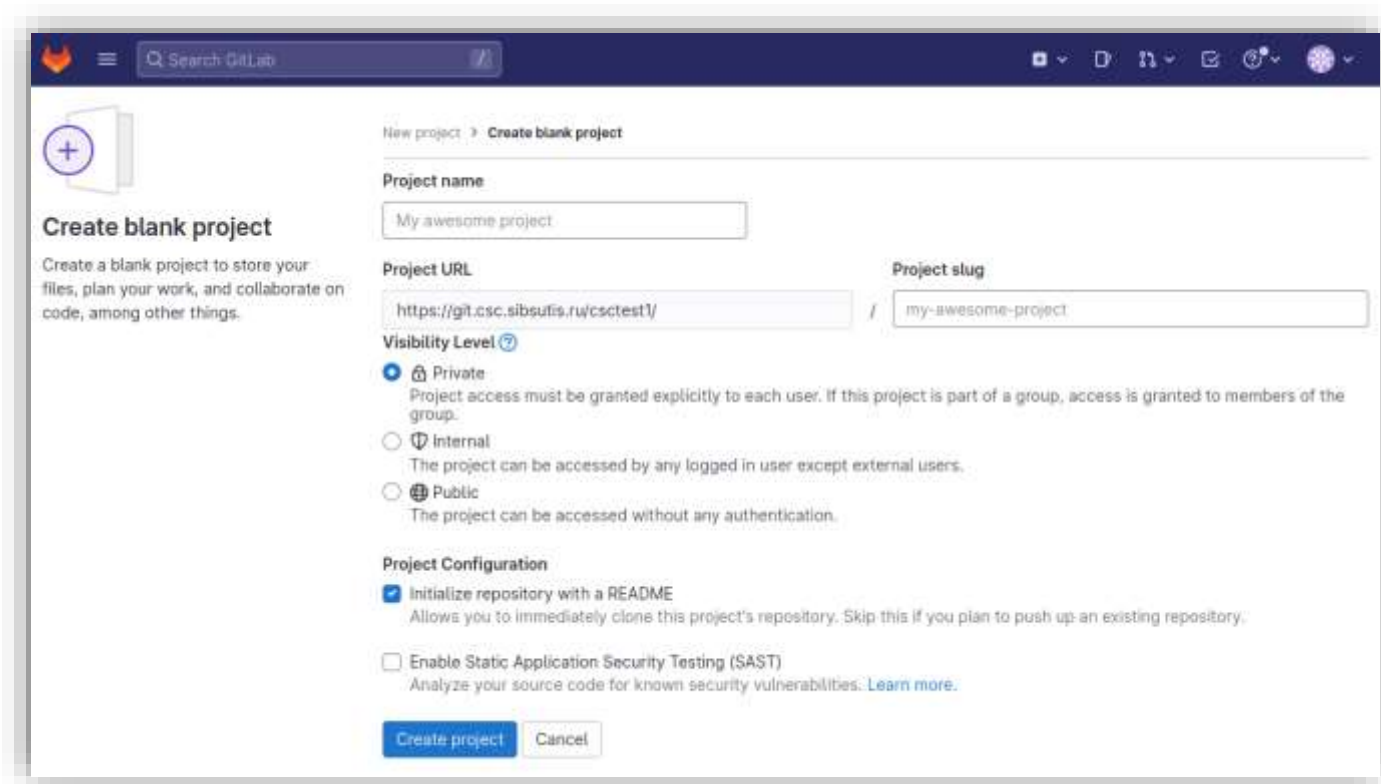


Рисунок 3 - Страница с выбором способа создания репозитория

Путь к репозиторию формируется автоматически на основе предлагаемого имени проекта. Его можно при необходимости откорректировать, но чаще всего это не обязательно.



*Рисунок 4 - Страница с начальными настройками репозитория*

Способ доступа определяет видимость вашего проекта другим пользователям системы хранения репозитория:

- Частный (private) режим дает доступ только тем пользователям, которые включены в команду разработки. По умолчанию в эту команду включен только создатель репозитория.
- Внутренний (internal) режим разрешает доступ к создаваемому репозиторию всем зарегистрированным пользователям системы хранения репозитория. По умолчанию режим доступа — только на чтение.
- Внешний (public) разрешает доступ к репозиторию всем.

Начальные настройки позволяют добавить в репозиторий файл README с типовым содержанием и настроить репозиторий на использование системы анализа кода на известные уязвимости при размещении коммитов.

После того, как все начальные параметры будут введены, необходимо нажать на кнопку «Создать репозиторий» и на этом создание репозитория завершено. В пустом репозитории (см. рисунок 5) система показывает дальнейшие действия, которые необходимо выполнить, чтобы начать заполнять репозиторий (инициализировать локальный репозиторий, сделать в нем начальный коммит и отправить этот коммит в систему хранения репозитория).

Для выполнения заданий по дисциплине «Архитектура ЭВМ» Вам необходимо создать пустой репозиторий (без файла README и без SAST). Доступ к репозиторию – «Частный» (private).

## **4. КАК ДОБАВИТЬ ПРЕПОДАВАТЕЛЯ В ЧЛЕНЫ ГРУППЫ РАЗРАБОТЧИКОВ?**

Над одним проектом могут работать несколько человек (иногда даже десятков человек и более). Система хранения репозитория отвечает за то, чтобы контролировать доступ к репозиториям и тем самым задавать каждому пользователю перечень доступных для него действий.

Список участников рабочей группы проекта можно посмотреть и настроить в меню Project Information -> Members. Меню располагается в левой части страницы с содержанием репозитория.

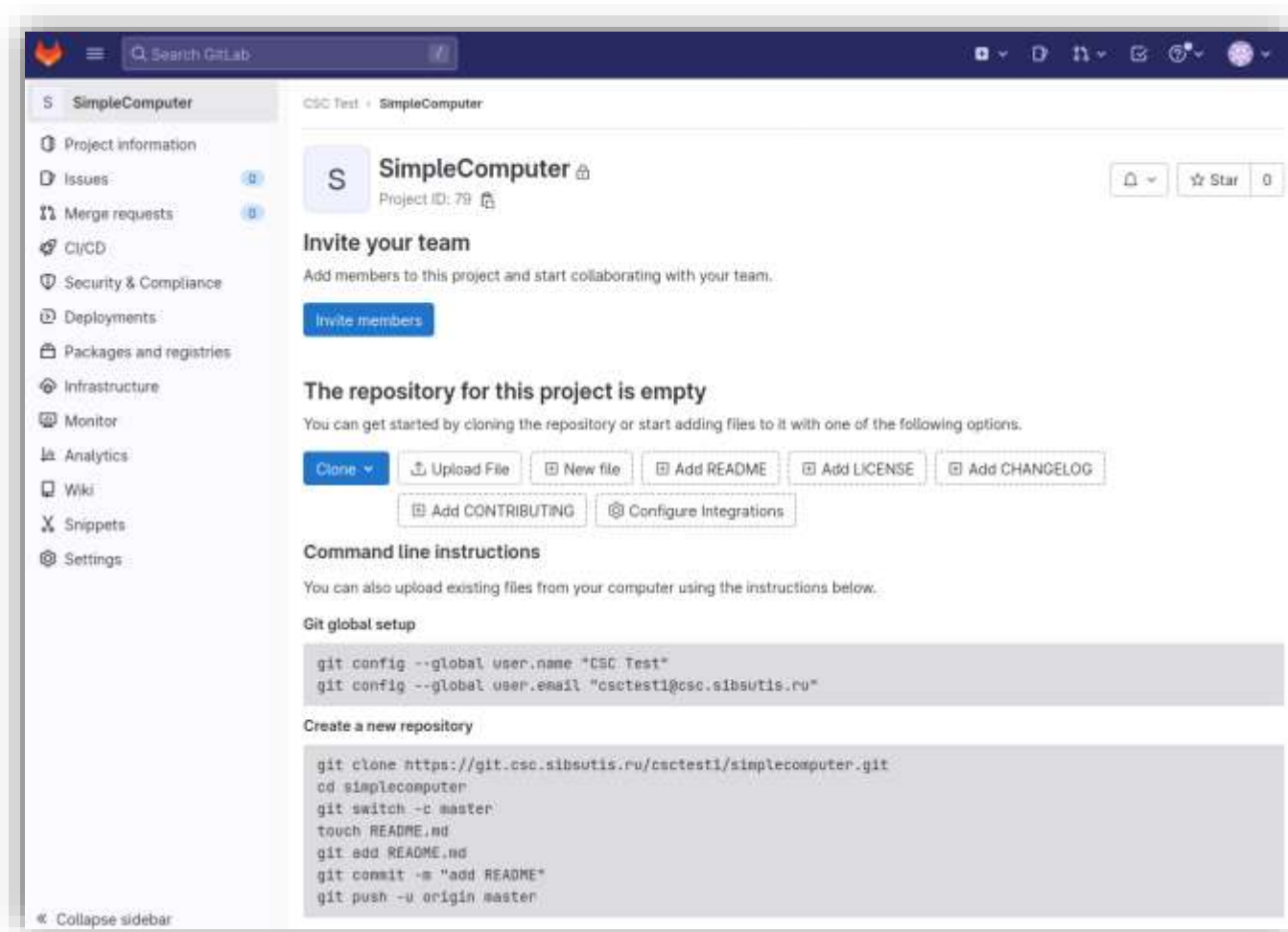


Рисунок 5 — Страница с пустым репозиторием

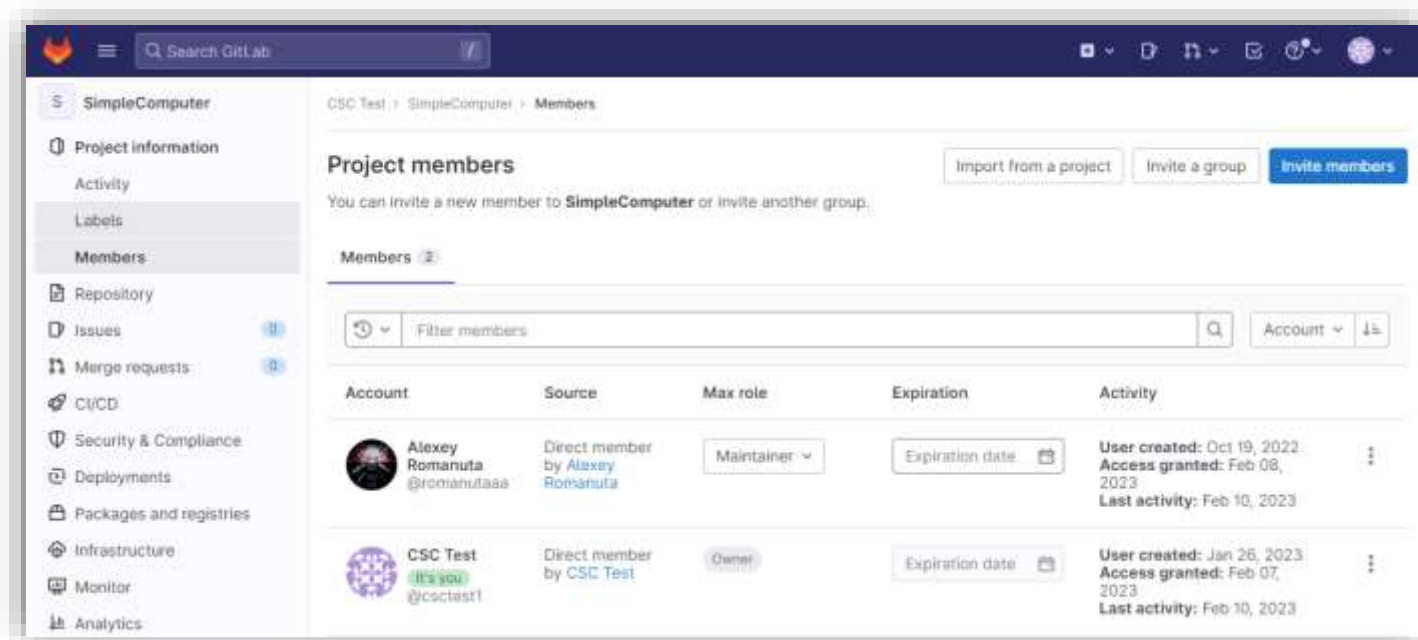


Рисунок 6 – Пример конфигурирование рабочей группы проекта

Чтобы добавить нового члена рабочей группы надо нажать на кнопку «Invite members» и дальше найти пользователь по его ФИО или нику зарегистрированному в системе **git.csc.sibsutis.ru**.

При добавлении каждому пользователю задается уровень доступа, который определяет что может делать пользователь с проектов и дата окончания его членства в рабочей группе.

Для сдачи заданий по дисциплине «Архитектура ЭВМ» Вам необходимо добавить преподавателя, ведущего практические занятия, со статусом не ниже «Reporter». Срок окончания членства преподавателя должен быть не ранее даты окончания семестра (можно дату оставить пустой, тогда преподаватель сам решит, когда покинут рабочую группу).

## 5. КАК СОЗДАТЬ ЛОКАЛЬНЫЙ РЕПОЗИТОРИЙ В ПУСТОМ КАТАЛОГЕ?

Для создания каталога с репозиторием **git** его необходимо скопировать (сленг. «клонировать») из системы хранения репозитория **git.csc.sibsutis.ru**. Для этого используется команда **git clone** в параметрах которой указывается ссылка на требуемый репозиторий. Пример такой операции представлен на рисунке 6.

```
[csctest1@jet tmp]$ git clone https://git.csc.sibsutis.ru/csctest1/simplecomputer.git
Клонирование в «simplecomputer»...
Username for 'https://git.csc.sibsutis.ru': csctest1
Password for 'https://csctest1@git.csc.sibsutis.ru':
warning: Похоже, что вы клонировали пустой репозиторий.
[csctest1@jet tmp]$ ls -la
итого 12
drwxr-xr-x 3 csctest1 csctest1 4096 фев 10 23:25 .
drwx----- 9 csctest1 csctest1 4096 фев 10 23:24 ..
drwxr-xr-x 3 csctest1 csctest1 4096 фев 10 23:25 simplecomputer
[csctest1@jet tmp]$ cd simplecomputer/
[csctest1@jet simplecomputer]$ ls -la
итого 12
drwxr-xr-x 3 csctest1 csctest1 4096 фев 10 23:25 .
drwxr-xr-x 3 csctest1 csctest1 4096 фев 10 23:25 ..
drwxr-xr-x 7 csctest1 csctest1 4096 фев 10 23:25 .git
[csctest1@jet simplecomputer]$
```

Рисунок 6 – Пример копирования репозитория из системы git.csc.sibsutis.ru

Ссылку на репозиторий можно найти на странице соответствующего проекта нажав на кнопку «Clone» (см. Рисунок 7). Система предложит Вам два способа доступа к репозиторию: с использованием протокола SSH и с использованием протокола HTTPS. Для доступа по протоколу SSH Ваш профиль должен быть предварительно настроен. Как это делается расскажем ниже, а пока получим доступ к репозиторию по протоколу HTTPS. В этой же «кнопке» система Вам предлагает открыть репозиторий сразу в некоторых популярных IDE. Для этого интегрированные среды должны быть определённым образом настроены, но это выходит за рамки данной шпаргалки.

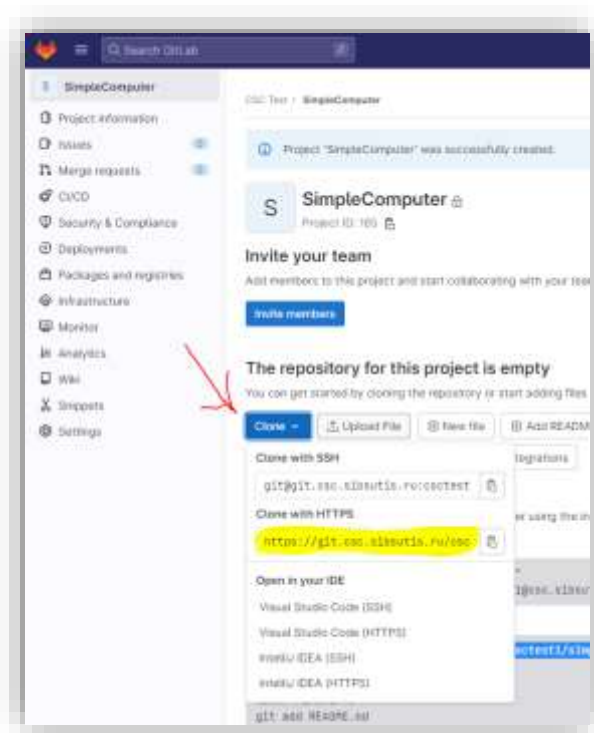


Рисунок 7 – Пример поиска ссылки на репозиторий в системе git.csc.sibsutis.ru

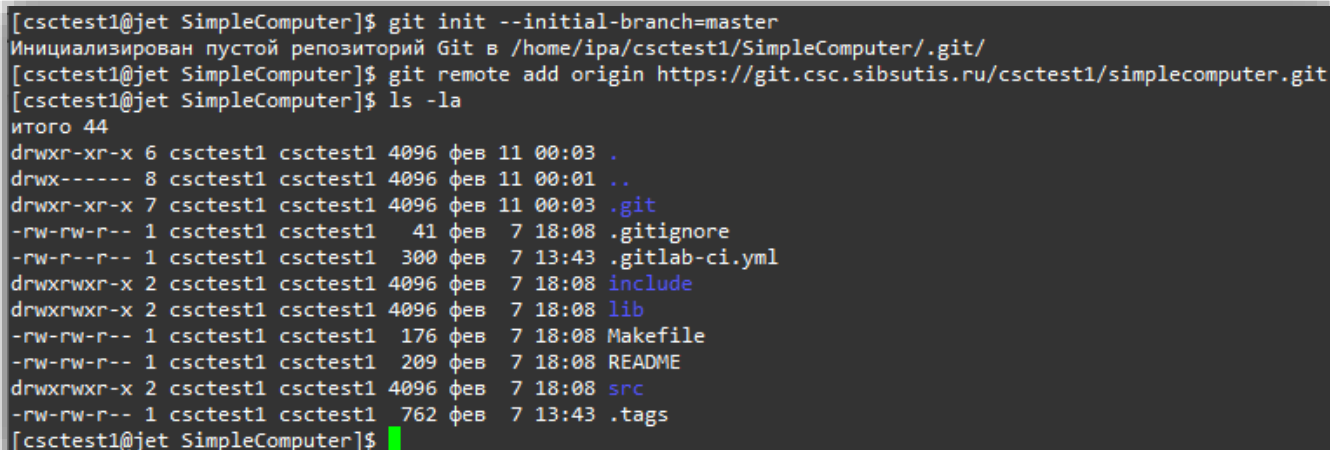


После клонирования репозитория в текущем каталоге создается папка, внутри которой будет локальный репозиторий `git` и файлы, восстановленные из этого репозитория из его основной ветки (либо каталог, если в репозитории ещё нет файлов). Если Вы хотите определить название создаваемого каталога, то его можно указать вторым параметром команды `git clone`. Клонировать репозиторий можно только во вновь создаваемый каталог.

## 6. КАК СОЗДАТЬ РЕПОЗИТОРИЙ В КАТАЛОГЕ С ИСХОДНЫМ КОДОМ ПРИЛОЖЕНИЯ?

В ситуации, когда репозиторий создается после того, как была начата разработка приложения, в каталоге с исходным кодом приложения необходимо инициализировать пустой репозиторий `git`. Делается это с помощью команды `git init`. В результате этой команды в текущем каталоге будет создан скрытый каталог `.git` и он будет проинициализирован так, чтобы в него можно было добавлять файлы и информацию об их изменениях. При создании можно указать имя основной ветки, добавив к команде опцию `-b` с указанием требуемого имени. По умолчанию создается ветка `main` (или `master`, в зависимости от версии используемого `git`). Инициализация пустого репозитория никак не влияет на расположенные в каталоге файлы.

После создания пустого репозитория его необходимо настроить указав ссылку на внешнее хранилище репозитория с помощью команды `git remote add`. Ссылка на репозиторий указывается точно так же, как в случае клонирования репозитория. «origin» - это локальное имя внешнего репозитория, с которым настраивается связь и которое будет дальше использоваться при операциях обмена информацией между локальным и внешним репозиториями.



```
[csctest1@jet SimpleComputer]$ git init --initial-branch=master
Инициализирован пустой репозиторий Git в /home/ipa/csctest1/SimpleComputer/.git/
[csctest1@jet SimpleComputer]$ git remote add origin https://git.csc.sibsutis.ru/csctest1/simplecomputer.git
[csctest1@jet SimpleComputer]$ ls -la
итого 44
drwxr-xr-x 6 csctest1 csctest1 4096 фев 11 00:03 .
drwx----- 8 csctest1 csctest1 4096 фев 11 00:01 ..
drwxr-xr-x 7 csctest1 csctest1 4096 фев 11 00:03 .git
-rw-rw-r-- 1 csctest1 csctest1 41 фев 7 18:08 .gitignore
-rw-r--r-- 1 csctest1 csctest1 300 фев 7 13:43 .gitlab-ci.yml
drwxrwxr-x 2 csctest1 csctest1 4096 фев 7 18:08 include
drwxrwxr-x 2 csctest1 csctest1 4096 фев 7 18:08 lib
-rw-rw-r-- 1 csctest1 csctest1 176 фев 7 18:08 Makefile
-rw-rw-r-- 1 csctest1 csctest1 209 фев 7 18:08 README
drwxrwxr-x 2 csctest1 csctest1 4096 фев 7 18:08 src
-rw-rw-r-- 1 csctest1 csctest1 762 фев 7 13:43 .tags
[csctest1@jet SimpleComputer]$
```

Рисунок 7 — Пример создания пустого локального репозитория

## 7. КАК СОЗДАТЬ НАЧАЛЬНЫЙ КОММИТ В ОСНОВНОЙ ВЕТКЕ РЕПОЗИТОРИЯ?

После того, как Вы создали локальный репозиторий, необходимо в основной ветке проекта создать хотя бы одну запись, содержащую информацию о каком-либо файле<sup>1</sup>. Для этого в каталоге, где располагается репозиторий, необходимо создать добавляемый файл (или этот файл уже есть, если создание репозитория было сделано после начала разработки). После этого файл должен быть добавлен в репозиторий и это состояние репозитория зафиксировано. Зафиксированное состояние должно быть отправлено в систему хранения репозитория `git.csc.sibsutis.ru`. Добавлять в первый раз можно не все файлы (например только файл `README`).

<sup>1</sup> Если при создании репозитория в системе `git.csc.sibsutis.ru` Вы выбрали автоматическое создание файла `README`, то система хранения репозитория выполнила этот шаг вместо Вас и репозиторий уже содержит первоначальный файл. В этом случае локальный репозиторий должен создаваться только путем его клонирования из системы хранения репозитория и создавать первую запись уже нет необходимости.

## 8. КАК ДОБАВИТЬ ФАЙЛЫ В ЛОКАЛЬНЫЙ РЕПОЗИТОРИЙ?

Цикл работы с локальным репозиторием как правило состоит из двух этапов: подготовка данных (добавление файлов), фиксация состояния репозитория (создание коммита, в котором записывается информация обо всех добавленных файлах или об их изменениях).

Для добавления файлов в локальный репозиторий используется команда **git add <путь>**, где **<путь>** - это имя файла (или каталога с файлами) с указанием пути от корневого каталога проекта. Например, **git add file.txt** — добавит файл **file.txt** в репозиторий. Команда **git add src/file.txt** — добавит файл **file.txt** из каталога **src**. Команда **git add src/** - добавит все файлы и каталоги, расположенные в каталоге **src**. В указании пути допустимо использовать традиционные символы масок файлов (\*, ?, []), а также символ «.», который означает, что необходимо добавить информацию об изменении всех файлов в указанном каталоге и его подкаталогах.

После добавления файла в репозиторий создается запись, отражающая изменения этого файла по отношению к предыдущему состоянию репозитория. Если добавляется новый файл, то предыдущем считается состояние, в котором этого файла нет, а его изменением будет текущее содержание файла.

```
[csctest1@jet SimpleComputer]$ git add .
[csctest1@jet SimpleComputer]$ git status
На ветке main

Еще нет коммитов

Изменения, которые будут включены в коммит:
(используйте «git rm --cached <файл>...», чтобы убрать из индекса)
    новый файл:   .gitignore
    новый файл:   Makefile
    новый файл:   README
    новый файл:   include/simplecomputer.h
    новый файл:   lib/Makefile
    новый файл:   lib/memory.c
    новый файл:   src/Makefile

[csctest1@jet SimpleComputer]$
```

Рисунок 8 — Пример добавления информации об изменении файлов в рабочем каталоге

## 9. КАК ПОСМОТРЕТЬ ТЕКУЩЕЕ СОСТОЯНИЕ ЛОКАЛЬНОГО РЕПОЗИТОРИЯ?

Чтобы посмотреть состояние репозитория в сравнении с файлами самого приложения используется команда **git status**. В результате будет выведена информация о последнем коммите текущей ветки и состоянии файлов с исходным кодом приложения по сравнению с тем, что хранится в локальном репозитории: файлы были изменены (удалены) и их изменения подготовлены для фиксации, файлы, которые были изменены, но их изменения не были включены в репозиторий для фиксации, файлы были удалены, появились новые не отслеживаемые файлы и т.п. С помощью этой информации разработчик легко может увидеть, какие файлы изменились в исходном коде приложения по сравнению с тем, что было зафиксировано ранее в локальном репозитории. Чтобы посмотреть какие именно изменения файлов были выполнены используется команда **git diff**.



```

[csctest1@jet SimpleComputer]$ git add .gitignore Makefile
[csctest1@jet SimpleComputer]$ git status
На ветке main

Еще нет коммитов

Изменения, которые будут включены в коммит:
(используйте «git rm --cached <файл>...», чтобы убрать из индекса)
    новый файл:   .gitignore
    новый файл:   Makefile

Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
    README
    include/
    lib/
    src/

[csctest1@jet SimpleComputer]$

```

Рисунок 7 — Пример вывода состояния рабочего каталога приложения.

## 10. КАК ЗАФИКСИРОВАТЬ («ЗАКОММИТИТЬ») ИНФОРМАЦИЮ О ДОБАВЛЕННЫХ ФАЙЛАХ (ИХ ИЗМЕНЕНИИ)?

После того, как Вы добавили информацию обо всех файлах, состояние которых Вам необходимо отразить в репозитории, необходимо зафиксировать созданные записи. Такая фиксация называется коммитом (от англ. слова commit) и выполняется она с помощью команды **git commit**. Любая фиксация состояния репозитория снабжается текстовым комментарием, содержащим краткую информацию о проделанных изменениях. По умолчанию **git** запустит для Вас текстовый редактор, в котором Вам необходимо будет ввести комментарий. Если Вы не хотите, чтобы запускался редактор, то комментарий можно указать в командной строке с помощью опции **-m**.

```

[csctest1@jet SimpleComputer]$ git add .
[csctest1@jet SimpleComputer]$ git commit -m "First commit"
[main (корневой коммит) ada40a4] First commit
7 files changed, 92 insertions(+)
create mode 100644 .gitignore
create mode 100644 Makefile
create mode 100644 README
create mode 100644 include/simplecomputer.h
create mode 100644 lib/Makefile
create mode 100644 lib/memory.c
create mode 100644 src/Makefile
[csctest1@jet SimpleComputer]$ git log
commit ada40a4264390f1548805b5b6f3eead86ac460fc (HEAD -> main)
Author: CSC Test <csctest1@jet.csc.sibsutis.ru>
Date: Sat Feb 4 18:41:34 2023 +0700

    First commit
[csctest1@jet SimpleComputer]$

```

Рисунок 9 — Пример фиксации (коммита) изменений в рабочем каталоге

## 11. КАК ОТПРАВИТЬ КОММИТЫ ИЗ ЛОКАЛЬНОГО РЕПОЗИТОРИЯ В ХРАНИЛИЩЕ GIT.CSC.SIBSUTIS.RU?

Отправка состояния локального репозитория во внешний репозиторий осуществляется с помощью команды **git push**. Если эта отправка является первой, т.е. ранее ни в локальном, ни во внешнем репозитории не было никакой информации, то в параметрах команды необходимо уточнить куда мы хотим отправить состояние нашего репозитория и как должна называться ветка в удалённом репозитории, в которую мы хотим отправить наши изменения. Пример такой команды представлен на Рисунке 10.

```
[csctest1@jet SimpleComputer]$ git log --oneline
efa86f0 (HEAD -> master) Начальный коммит
[csctest1@jet SimpleComputer]$ git push -u origin master
Username for 'https://git.csc.sibsutis.ru': csctest1
Password for 'https://csctest1@git.csc.sibsutis.ru':
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 397 байтов | 397.00 КиБ/с, готово.
Всего 3 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
To https://git.csc.sibsutis.ru/csctest1/simplecomputer.git
 * [new branch]      master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
[csctest1@jet SimpleComputer]$
```

Рисунок 10 – Пример отправки состояния локального репозитория во внешний.

## 12. КАК СОЗДАТЬ НОВУЮ ВЕТКУ ДЛЯ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОГО ЗАДАНИЯ?

По требованиям выполнения заданий в дисциплине «Архитектура ЭВМ» каждое практическое задание должно выполняться в отдельной ветке репозитория и после защиты «соединяться» с основной веткой.

Для создания ветки необходимо использовать команду **git switch** с опцией **-c**. Посмотреть список существующих веток и какая из этих веток является текущей (т.е. в неё будут добавляться файлы и делаться коммиты) можно с помощью команды **git branch**.

```
[csctest1@jet SimpleComputer]$ git switch -c pract01
Переключено на новую ветку «pract01»
[csctest1@jet SimpleComputer]$ git branch
  master
* pract01
[csctest1@jet SimpleComputer]$ git status
На ветке pract01
Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
.gitignore
.gitlab-ci.yml
Makefile
include/
lib/
src/

ничего не добавлено в коммит, но есть неотслеживаемые файлы (используйте «git add», чтобы отслеживать их)
[csctest1@jet SimpleComputer]$
```

Рисунок 11 – пример создания новой ветки.