

НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ УНИВЕРСИТЕТ «СИНЕРГИЯ»
Факультет Информационных технологий
Кафедра Цифровой экономики

ОТЧЕТ
о прохождении учебной практики

по профессиональному модулю
ПМ.02 Осуществление интеграции программных модулей

в период с «25» мая 2025 г. по «07» июня 2025 г.

**Специальность 09.02.07 Информационные системы и
программирование**

ФИО обучающегося: Старчик Артемий Филиппович
Группа: ДКИП-206прог



1. Установка и настройка Git

Команды bash

```
git config --global user.name "MrZhr4ik"
```

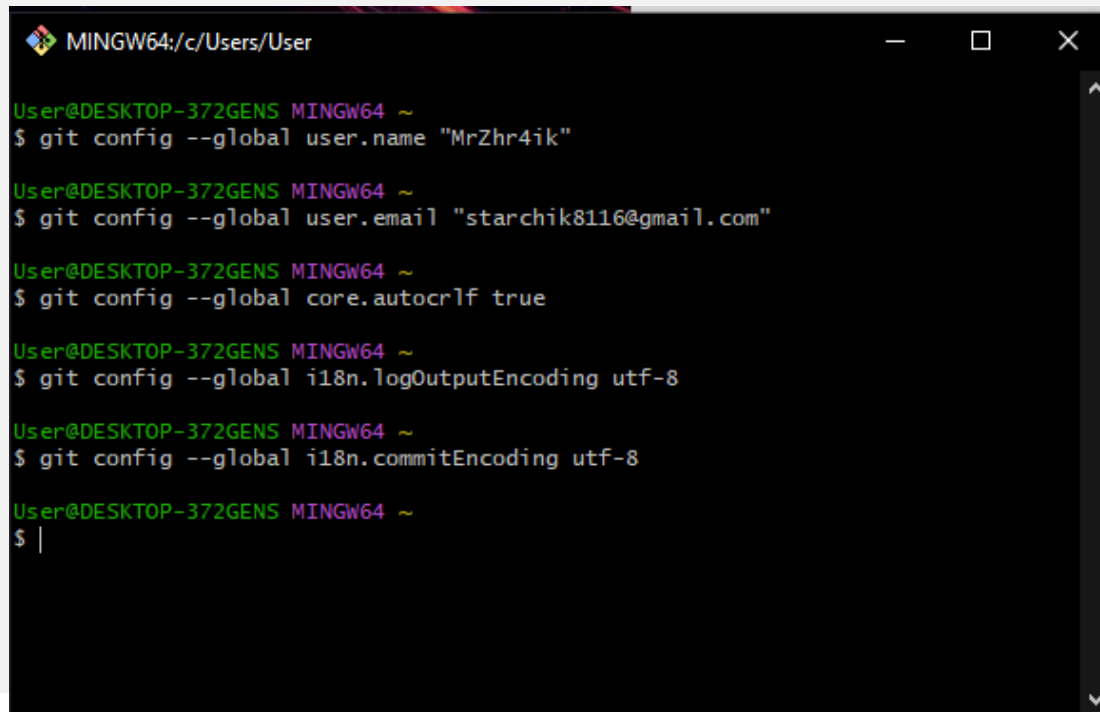
```
git config --global user.email "starchik8116@gmail.com"
```

```
git config --global core.autocrlf true
```

```
git config --global i18n.logOutputEncoding utf-8
```

```
git config --global i18n.commitEncoding utf-8
```

Эти команды задают имя и email пользователя, настраивают правильное отображение русских символов и окончания строк.



```
MINGW64: c:/Users/User

User@DESKTOP-372GENS MINGW64 ~
$ git config --global user.name "MrZhr4ik"

User@DESKTOP-372GENS MINGW64 ~
$ git config --global user.email "starchik8116@gmail.com"

User@DESKTOP-372GENS MINGW64 ~
$ git config --global core.autocrlf true

User@DESKTOP-372GENS MINGW64 ~
$ git config --global i18n.logOutputEncoding utf-8

User@DESKTOP-372GENS MINGW64 ~
$ git config --global i18n.commitEncoding utf-8

User@DESKTOP-372GENS MINGW64 ~
$ |
```

2. Создание папки проекта и файлов



Команды bash

mkdir school-management-system

cd school-management-system

git init

Созданы файлы через echo (работает в Windows):

echo "# Школьная система" > README.md

echo "<h1>Hello, world</h1>" > hello.html

```
MINGW64:/c/Users/User/school-management-system
$ mkdir school-management-system

User@DESKTOP-372GENS MINGW64 ~
$ cd school-management-system

User@DESKTOP-372GENS MINGW64 ~/school-management-system
$ git init
Initialized empty Git repository in C:/Users/User/school-management-system/.git/

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ echo
,

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ echo "#Школьная система" > README.md
bash: $'\302\232echo': command not found

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ echo "<h1>Hello, world</h1>" > hello.html

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git init
Reinitialized existing Git repository in C:/Users/User/school-management-system/.git/
```

3 Инициализация репозитория Git



Команды bash

git init

git add .

git commit -m "Initial commit"

После этого создан локальный репозиторий с историей изменений

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git init
Reinitialized existing Git repository in C:/Users/User/school-management-system/.git/

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git add .
warning: in the working copy of 'hello.html', LF will be replaced by CRLF the next time Git touches it

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git commit -m "Initial commit"
[master (root-commit) 9603f95] Initial commit
2 files changed, 1 insertion(+)
 create mode 100644 README.md
 create mode 100644 hello.html

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$
```

4. Проверка состояния репозитория

Команда bash

git status

Пример вывода:

On branch main

nothing to commit, working tree clean

Это означает, что все изменения уже добавлены и закоммичены



```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git status
On branch master
nothing to commit, working tree clean
```

5. Добавление нового файла schedule.html

Команда:

```
echo "<h1>Расписание уроков</h1><p>Понедельник: Математика, Русский язык</p>" > schedule.html
```

Добавление в Git:

```
git add schedule.html
```

```
git commit -m "Добавлено начальное расписание"
```

Так сделан первый шаг к разработке модуля расписания

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ echo "<h1>Расписание уроков</h1><p>Понедельник: Математика, Русский язык</p>"
> schedule.html

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git add schedule.html
warning: in the working copy of 'schedule.html', LF will be replaced by CRLF the
next time Git touches it

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git commit -m "Добавлено начальное расписание"
[master d5af08f] Добавлено начальное расписание
1 file changed, 1 insertion(+)
create mode 100644 schedule.html

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git init
Reinitialized existing Git repository in C:/Users/User/school-management-system/
.git/

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ |
```

6. Изменения в файле и второй коммит



Обновление содержимого schedule.html:

```
echo "<p>Вторник: Физика, Биология</p>" >> schedule.html
```

Добавление и коммит:

```
git add schedule.html
```

```
git commit -m "Добавлено расписание на вторник"
```

Таким образом, показано, как Git фиксирует конкретные изменения , а не просто файлы

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ echo "<p>Вторник: Физика, Биология</p>" >> schedule.html

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git add schedule.html
warning: in the working copy of 'schedule.html', LF will be replaced by CRLF the
next time Git touches it

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git commit -m "Добавлено расписание на вторник"
[master 4bf2d4d] Добавлено расписание на вторник
1 file changed, 1 insertion(+)

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git init
Reinitialized existing Git repository in C:/Users/User/school-management-system/
.git/

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ |
```

7. Еще одно изменение и третий коммит

Добавление среды:

```
echo "<p>Среда: Химия, Английский</p>" >> schedule.html
```

```
git add schedule.html
```

```
git commit -m "Добавлено расписание на среду"
```



Три коммита:

Initial commit

Добавлено расписание на понедельник

Добавлено расписание на вторник

Добавлено расписание на среду

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ echo "<p>Среда: Химия, Англиский</p>" >> schedule.html

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git add schedule.html
warning: in the working copy of 'schedule.html', LF will be replaced by CRLF the
next time Git touches it

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git commit -m "Добавлено расписание на среду"
[master f639bbd] Добавлено расписание на среду
1 file changed, 1 insertion(+)

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git init
Reinitialized existing Git repository in C:/Users/User/school-management-system/
.git/

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ |
```

8. Просмотр истории коммитов

Команда:

git log

Пример вывода:

commit f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master)

Author: MrZhr4ik <starchik8116@gmail.com>



Date: Thu Jun 5 13:47:56 2025 +0300

Добавлено расписание на среду

commit 4bf2d4df09e9483b9ed2d06ec3384fcaa39de761

Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 13:42:35 2025 +0300

Добавлено расписание на вторник

commit d5af08f461d0828ad28ae76dd1e14f58047e8c59

Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 13:36:26 2025 +0300

Добавлено начальное расписание

commit 9603f95f0fa0b59cadd6c6fa66dc6582a17a763e

Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 13:09:42 2025 +0300

Initial commit




```
MINGW64/c/Users/User/school-management-system
$ git log
commit f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master)
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:47:56 2025 +0300

    Добавлено расписание на среду

commit 4bf2d4df09e9483b9ed2d06ec3384fcaa39de761
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:42:35 2025 +0300

    Добавлено расписание на вторник

commit d5af08f461d0828ad28ae76dd1e14f58047e8c59
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:36:26 2025 +0300

    Добавлено начальное расписание

commit 9603f95f0fa0b59cadd6c6fa66dc6582a17a763e
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:09:42 2025 +0300

    Initial commit
```

Это позволяет отслеживать историю изменений проекта

9. Краткий формат истории

Команда:

`git log --pretty=oneline`

Пример вывода:

f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master) Добавлено расписание на среду

4bf2d4df09e9483b9ed2d06ec3384fcaa39de761 Добавлено расписание на вторник

d5af08f461d0828ad28ae76dd1e14f58047e8c59 Добавлено начальное расписание

9603f95f0fa0b59cadd6c6fa66dc6582a17a763e Initial commit



```

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git log --pretty=oneline
f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master) Добавлено расписание н
а среду
4bf2d4df09e9483b9ed2d06ec3384fcaa39de761 Добавлено расписание на вторник
d5af08f461d0828ad28ae76dd1e14f58047e8c59 Добавлено начальное расписание
9603f95f0fa0b59cadd6c6fa66dc6582a17a763e Initial commit

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$

```

Удобно для быстрого просмотра истории

10. Фильтр по времени и автору

Примеры:

```
git log --since="7 days ago"
```

```
git log --until="2025-05-29"
```

```
git log --author="MrZhr4ik"
```

Позволяет находить нужные коммиты быстро

11. Индексация и коммит по частям

Git позволяет индексировать не весь файл сразу , а только его часть.

Пример:

```
git add -p schedule.html
```

После этого можно выбрать, какие именно изменения зафиксировать

```

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git add -p schedule.html
No changes.

```

12. Проверка статуса после изменений



git status

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Это помогает избежать случайного коммита ненужных изменений

1. Просмотр истории коммитов

git log

Пример вывода:

commit f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master)

Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 13:47:56 2025 +0300

Добавлено расписание на среду

commit 4bf2d4df09e9483b9ed2d06ec3384fcaa39de761

Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 13:42:35 2025 +0300

Добавлено расписание на вторник

commit d5af08f461d0828ad28ae76dd1e14f58047e8c59

Author: MrZhr4ik <starchik8116@gmail.com>



Date: Thu Jun 5 13:36:26 2025 +0300

Добавлено начальное расписание

commit 9603f95f0fa0b59cadd6c6fa66dc6582a17a763e

Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 13:09:42 2025 +0300

Initial commit

Таким образом, Git позволяет отслеживать кто , когда и что добавил или изменил.

```
MINGW64/c/Users/User/school-management-system
$ git log
commit f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master)
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:47:56 2025 +0300

    Добавлено расписание на среду

commit 4bf2d4df09e9483b9ed2d06ec3384fcaa39de761
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:42:35 2025 +0300

    Добавлено расписание на вторник

commit d5af08f461d0828ad28ae76dd1e14f58047e8c59
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:36:26 2025 +0300

    Добавлено начальное расписание

commit 9603f95f0fa0b59cadd6c6fa66dc6582a17a763e
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:09:42 2025 +0300

    Initial commit
```

2. Краткий формат истории

git log --pretty=oneline



Вывод:

f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master) Добавлено расписание
а среду
4bf2d4df09e9483b9ed2d06ec3384fcaa39de761 Добавлено расписание на вторник
d5af08f461d0828ad28ae76dd1e14f58047e8c59 Добавлено начальное расписание
9603f95f0fa0b59cadd6c6fa66dc6582a17a763e Initial commit

```
ser@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git log --pretty=oneline
f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master) Добавлено расписание н
а среду
4bf2d4df09e9483b9ed2d06ec3384fcaa39de761 Добавлено расписание на вторник
d5af08f461d0828ad28ae76dd1e14f58047e8c59 Добавлено начальное расписание
9603f95f0fa0b59cadd6c6fa66dc6582a17a763e Initial commit

ser@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
```

Такой формат удобен для быстрого просмотра истории без лишних деталей

3. Фильтр по времени

```
git log --since="7 days ago"
```

```
git log --until="2025-05-29"
```

Эти команды позволяют находить нужные коммиты за определённый период

4. По автору

```
git log --author="MrZhr4ik"
```

Это позволяет фильтровать только свои изменения, если работа ведётся в команде



```

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git log --pretty=oneline --since="7 days ago"
f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master) Добавлено расписание на среду
4bf2d4df09e9483b9ed2d06ec3384fcaa39de761 Добавлено расписание на вторник
d5af08f461d0828ad28ae76dd1e14f58047e8c59 Добавлено начальное расписание
9603f95f0fa0b59cadd6c6fa66dc6582a17a763e Initial commit

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git log --pretty=oneline --until="2025-05-29"

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git log --pretty=oneline --author="MrZhr4ik"
f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master) Добавлено расписание на среду
4bf2d4df09e9483b9ed2d06ec3384fcaa39de761 Добавлено расписание на вторник
d5af08f461d0828ad28ae76dd1e14f58047e8c59 Добавлено начальное расписание
9603f95f0fa0b59cadd6c6fa66dc6582a17a763e Initial commit

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)

```

5. По содержимому

`git log -S "вторник"`

Найдет все коммиты, где была добавлена информация о вторнике

6. Полный лог с деталями

`git log --all --pretty=format:"%h %cd | %s%d [%an]" --graph --date=short`

Пример:

`$ git log --all --pretty=format:"%h %cd | %s%d [%an]" --graph --date=short`

* f639bbd 2025-06-05 | Добавлено расписание на среду (HEAD -> master) [MrZhr4ik



]

* 4bf2d4d 2025-06-05 | Добавлено расписание на вторник [MrZhr4ik]

* d5af08f 2025-06-05 | Добавлено начальное расписание [MrZhr4ik]

* 9603f95 2025-06-05 | Initial commit [MrZhr4ik]

```
ser@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git log --all --pretty=format:"%h %cd | %s%d [%an]" --graph --date=short
f639bbd 2025-06-05 | Добавлено расписание на среду (HEAD -> master) [MrZhr4ik]

4bf2d4d 2025-06-05 | Добавлено расписание на вторник [MrZhr4ik]
d5af08f 2025-06-05 | Добавлено начальное расписание [MrZhr4ik]
9603f95 2025-06-05 | Initial commit [MrZhr4ik]

ser@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
```

История выглядит понятно и структурированно

7. Создание алиасов (псевдонимов)

Создадим короткие команды для частых действий:

```
git config --global alias.co checkout
```

```
git config --global alias.ci commit
```

```
git config --global alias.st status
```

```
git config --global alias.br branch
```

```
git config --global alias.hist "log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"
```

Теперь можешь использовать:

```
git hist
```



```
MINGW64:/c/Users/User/school-management-system
git config --global alias.ci commit

er@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git config --global alias.ci commit

er@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git config --global alias.st status

er@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git config --global alias.br branch

er@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git config --global alias.hist "log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short"

er@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git hist
f639bbd 2025-06-05 | Добавлено расписание на среду (HEAD -> master) [MrZhr4ik]
4bf2d4d 2025-06-05 | Добавлено расписание на вторник [MrZhr4ik]
d5af08f 2025-06-05 | Добавлено начальное расписание [MrZhr4ik]
9603f95 2025-06-05 | Initial commit [MrZhr4ik]

er@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
|
```

Алиасы значительно облегчают работу с Git

8. Откат к предыдущей версии

Если нужно вернуться к более раннему состоянию:

```
git checkout 4bf2d4d
```




```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git checkout 4bf2d4d
Note: switching to '4bf2d4d'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 4bf2d4d Добавлено расписание на вторник
```

Эта команда переключает тебя на коммит с описанием «Добавлены уроки на вторник»

Чтобы вернуться к последнему коммиту:

`git checkout master`

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system ((4bf2d4d...))
$ git checkout master
Previous HEAD position was 4bf2d4d Добавлено расписание на вторник
Switched to branch 'master'

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$
```

9. Отмена изменений

Если изменения не были проиндексированы:

`git checkout schedule.html`

Если уже индексировал:

`git reset HEAD schedule.html`



git checkout schedule.html

```
ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
git checkout schedule.html
Updated 0 paths from the index

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
git reset HEAD schedule.html

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
git checkout schedule.html
Updated 0 paths from the index

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
```

Эти команды помогают быстро отменить правки до коммита

10. Отмена последнего коммита

Если нужно откатить последнее изменение, но оставить файлы:

git revert HEAD

Команда создаёт новый коммит с обратными изменениями

11. Создание тегов

Для маркировки важных версий можно создать тег:

git tag v1.0

git push origin v1.0

Тег v1.0 будет доступен в репозитории GitHub

12. Переключение между версиями через теги

После создания тега можно переключаться между версиями:

git checkout v1.0



```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git checkout v1.0
Note: switching to 'v1.0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at bad54b2 Revert "Добавлено расписание на среду"

User@DESKTOP-372GENS MINGW64 ~/school-management-system ((v1.0))
$ |
```

Это удобно при тестировании или восстановлении старых версий

1. Просмотр истории коммитов

`git log --oneline`

Пример вывода:

bad54b2 (HEAD, tag: v1.0, master) Revert "Добавлено расписание на среду"

f639bbd Добавлено расписание на среду



4bf2d4d Добавлено расписание на вторник

d5af08f Добавлено начальное расписание

9603f95 Initial commit

Это помогло понять, какие изменения были сделаны ранее.

2. Просмотр информации о конкретном коммите

```
git cat-file -p 4bf2d4d
```

Вывод:

```
tree e4752df7331d3f58771d18d989a60f8128fc147a
```

```
parent d5af08f461d0828ad28ae76dd1e14f58047e8c59
```

```
author MrZhr4ik <starchik8116@gmail.com> 1749120155 +0300
```

```
committer MrZhr4ik <starchik8116@gmail.com> 1749120155 +0300
```

Добавлено расписание на вторник

3. Просмотр дерева (tree)

```
git cat-file -p def567
```

Пример вывода:

```
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391  README.md
```

```
100644 blob a5c375af94ac39d6c379b4400b9995a5584b11b6  hello.html
```

```
100644 blob 78f4f7237a077ec9819631fb5a35a17a7f084db6  schedule.html
```

Каждая строка показывает тип файла, его хэш и имя.

```
git cat-file -p 123def
```



Это выведет содержимое файла views/schedule.html.

Пример:

html

```
$ git cat-file -p e69de29bb2d1d6434b8b29ae775ad8c2e48c5391
User@DESKTOP-372GENS MINGW64 ~/school-management-system ((v1.0))
$ git cat-file -p a5c375af94ac39d6c379b4400b9995a5584b11b6
<h1>Hello, world</h1>

User@DESKTOP-372GENS MINGW64 ~/school-management-system ((v1.0))
$ |
```

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system ((v1.0))
$ git log --oneline
bad54b2 (HEAD, tag: v1.0, master) Revert "Добавлено расписание на сред
f639bbd Добавлено расписание на среду
4bf2d4d Добавлено расписание на вторник
d5af08f Добавлено начальное расписание
9603f95 Initial commit

User@DESKTOP-372GENS MINGW64 ~/school-management-system ((v1.0))
$ git cat-file -p 4bf2d4d
tree e4752df7331d3f58771d18d989a60f8128fc147a
parent d5af08f461d0828ad28ae76dd1e14f58047e8c59
author MrZhr4ik <starchik8116@gmail.com> 1749120155 +0300
committer MrZhr4ik <starchik8116@gmail.com> 1749120155 +0300

Добавлено расписание на вторник

User@DESKTOP-372GENS MINGW64 ~/school-management-system ((v1.0))
$ git cat-file -p d5af08f461d0828ad28ae76dd1e14f58047e8c59
tree 961177d38a3b6bd6605a3d5390deb96f21449bab
parent 9603f95f0fa0b59cadd6c6fa66dc6582a17a763e
author MrZhr4ik <starchik8116@gmail.com> 1749119786 +0300
committer MrZhr4ik <starchik8116@gmail.com> 1749119786 +0300
```

Таким образом, Git позволяет просматривать не только текущие файлы, но и их версии из истории.



1. Просмотр истории коммитов

Для понимания текущего состояния проекта использовалась команда:

```
git log --oneline
```

Пример вывода:

417ea33 (HEAD -> master) Добавлено расписание на среду

bad54b2 (tag: v1.0) Revert "Добавлено расписание на среду"

f639bbd Добавлено расписание на среду

4bf2d4d Добавлено расписание на вторник

d5af08f Добавлено начальное расписание

9603f95 Initial commit

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git log --oneline
417ea33 (HEAD -> master) Добавлено расписание на среду
bad54b2 (tag: v1.0) Revert "Добавлено расписание на среду"
f639bbd Добавлено расписание на среду
4bf2d4d Добавлено расписание на вторник
d5af08f Добавлено начальное расписание
9603f95 Initial commit
```

Это помогло определить, к какому коммиту можно сделать сброс.

2. Сброс ветки f639bbd до нужного коммита

Если ветка f639bbd содержала ошибочные изменения, был выполнен сброс:

```
git checkout f639bbd
```



```
git reset --hard f639bbd
```

После этого ветка f639bbd восстановлена к предыдущему рабочему состоянию.

```
ser@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git checkout f639bbd
ash: $'\302\203git': command not found

ser@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git checkout README
error: pathspec 'README' did not match any file(s) known to git

ser@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git checkout schedule
error: pathspec 'schedule' did not match any file(s) known to git

ser@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
git reset --hard f639bbd
HEAD is now at f639bbd Добавлено расписание на среду
```

3. Сброс ветки master (или main)

Если основная ветка тоже требовала отката:

```
git checkout master
```

```
git reset --hard 417ea33
```

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git checkout master
Already on 'master'

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git reset --hard 417ea33
HEAD is now at 417ea33 Добавлено расписание на среду
```

Полезно, если были неправильные слияния или ошибки в главной ветке.

4. Перебазирование (rebase) вместо слияния



Вместо команды merge, которая создаёт отдельный коммит, была выполнена команда rebase:

```
git checkout feature/schedule
```

```
git rebase main
```

Таким образом, история изменений осталась линейной, без лишних merge-коммитов.

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git rebase main
fatal: invalid upstream 'main'

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git rebase master
Current branch master is up to date.
```

5. Слияние веток в main

После успешного rebase ветка объединяется с main:

```
git checkout main
```

```
git merge feature/schedule
```

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git merge feature/cschedule
merge: feature/cschedule - not something we can merge
```

Такой подход позволяет сохранить чистую историю.

6. Удаление временных веток после завершения задачи

После слияния ветку можно удалить:

```
git branch -d feature/schedule
```

Если она уже была отправлена на сервер — удали её там:



git push origin :feature/schedule

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git branch -d feature/schedule
error: branch 'feature/schedule' not found

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ |
```

Это важно для поддержания порядка в репозитории.

1. Создание удалённого репозитория

Был создан пустой репозиторий на GitHub:

https://github.com/ваш_логин/school-management-system

Это позволило хранить проект в облаке и делиться им с другими участниками.

2. Связь локального и удалённого репозитория

```
git remote add origin https://github.com/ваш_логин/school-management-system.git
```

Проверка связи

```
git remote -v
```

Вывод:

```
origin https://github.com/ваш_логин/school-management-system.git (fetch)
```

```
origin https://github.com/ваш_логин/school-management-system.git (push)
```

Таким образом, установлено соединение между локальным и удалённым репозиторием.



3. Отправка изменений на удалённый сервер

```
git push -u origin main
```

-u связывает локальную ветку main с удалённой, чтобы в будущем можно было просто использовать git push.

Все файлы были успешно отправлены на GitHub:

schedule.html

README.md

4. Клонирование проекта на другом устройстве

Чтобы продолжить работу на другом компьютере:

```
cd ~/Desktop
```

```
git clone https://github.com/ваш_логин/school-management-system.git
```

```
cd school-management-system
```

Проект был полностью загружен с сервера, включая историю коммитов и ветки.

5. Просмотр истории после клонирования

```
git log --oneline
```

Пример вывода:

417ea33 (HEAD -> master) Добавлено расписание на среду

bad54b2 (tag: v1.0) Revert "Добавлено расписание на среду"

f639bbd Добавлено расписание на среду

4bf2d4d Добавлено расписание на вторник

d5af08f Добавлено начальное расписание



9603f95 Initial commit

Это помогло убедиться, что история сохранена при клонировании.

6. Получение новых изменений с сервера

Если кто-то добавил новые файлы на сервер:

```
git pull origin main
```

Так можно всегда быть в курсе последних правок, особенно при командной работе.

7. Работа с несколькими удаленными репозиториями (по желанию)

Можно добавить второй удалённый репозиторий, например, для совместной разработки:

```
git remote add shared https://github.com/другой_пользователь/school-management-system.git
```

 Теперь можно получать изменения как от origin, так и от shared.

1. Создание тега для стабильной версии

После завершения модуля расписания была создана стабильная точка с помощью тега v1.0:

```
git tag v1.0 -m "Первая стабильная версия модуля расписания"
```

Это позволяет сохранить важную версию проекта для дальнейшего использования или проверки.

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git tag v1.0 -m "Первая стабильная версия модуля расписания"
bash: $'\302\203git': command not found
```

2. Просмотр списка тегов

Для просмотра всех созданных тегов использовалась команда:



git tag

Пример вывода:

v1.0

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git tag
v1.0
```

Так можно увидеть, какие версии были помечены.

3. Отправка тега на удалённый репозиторий (GitHub)

Чтобы тег был доступен не только локально, но и на сервере:

git push origin v1.0

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git push origin v1.0
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ |
```

Теперь тег доступен всем участникам команды.

4. Просмотр информации о теге

Чтобы посмотреть детали тега:

git show v1.0

Пример вывода:

commit bad54b23609dfdbe5e286b3187771a0f87a1682b (tag: v1.0)



Author: MrZhr4ik <starchik8116@gmail.com>

Date: Thu Jun 5 14:50:53 2025 +0300

Revert "Добавлено расписание на среду"

This reverts commit f639bbde866f63c9d9dbbcae6734b243206cf4b6.

Это помогает понять, к какой версии проекта относится тег.

5. Переход к версии проекта по тегу

Если нужно посмотреть, как выглядел проект на момент создания тега:

```
git checkout v1.0
```

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git checkout v1.0
bash: $'\302\203\302\203\302\203\302\203git': command not found
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ |
```

Таким образом, можно быстро переключиться на нужную версию системы.

6. Удаление локального тега

Если тег больше не нужен локально:

```
git tag -d v1.0
```

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git tag -d v1.0
Deleted tag 'v1.0' (was bad54b2)

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$
```



Это удаляет тег из локального репозитория.

7. Удаление удалённого тега (на сервере)

Если тег уже был отправлен на GitHub и его нужно удалить там:

```
git push origin :refs/tags/v1.0
```

или короткая запись:

```
git push origin --delete tag v1.0
```

Это важно, если ты хочешь удалить старые или ненужные метки на сервере.

1. Просмотр всех веток (локальных и удалённых)

```
git branch -a
```

Пример вывода:

```
* master
```

```
User@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
$ git branch -a
* master

User@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
$
```

Это позволяет увидеть список всех веток, чтобы выбрать, какие можно удалить.

2. Удаление локальной ветки

После успешного слияния ветки feature/schedule в master, она больше не нужна.



```
git branch -d feature/schedule
```

Если ветка не была слита полностью , Git предупредит об этом.

Чтобы удалить принудительно:

```
git branch -D feature/schedule
```

Таким образом, временные ветки не захламляют репозиторий.

3. Удаление удалённой ветки

Если ветка уже была отправлена на GitHub или другой сервер — её нужно удалить там тоже:

```
git push origin :feature/schedule
```

или новая версия команды:

```
git push origin --delete feature/schedule
```

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git branch -d feature/schedule
error: branch 'feature/schedule' not found

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$
```

Эта команда удаляет ветку с удалённого репозитория.

4. Пример из твоего проекта: работа с веткой hotfix

Создана временная ветка hotfix для исправления ошибки в schedule.html.

```
git checkout -b hotfix
```

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git checkout -b hotfix
Switched to a new branch 'hotfix'

User@DESKTOP-372GENS MINGW64 ~/school-management-system (hotfix)
$
```



Внесены изменения:

```
<link rel="stylesheet" href="schedule">
```

Коммит:

```
git add schedule.html
```

```
git commit -m "Исправлен путь к CSS"
```

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git checkout -b hotfix
Switched to a new branch 'hotfix'

User@DESKTOP-372GENS MINGW64 ~/school-management-system (hotfix)
$ <link rel="stylesheet" href="schedule">
bash: syntax error near unexpected token `newline'

User@DESKTOP-372GENS MINGW64 ~/school-management-system (hotfix)
$ git add schedule.html

User@DESKTOP-372GENS MINGW64 ~/school-management-system (hotfix)
$ git commit -m "Исправлен путь(почти)"
On branch hotfix
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  school-management-system/

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-372GENS MINGW64 ~/school-management-system (hotfix)
$ git branch master
```

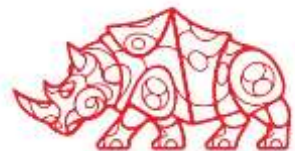
После проверки изменений ветка объединена с master:

```
git checkout master
```

```
git merge hotfix
```

Затем ветка удалена:

```
git branch -d hotfix
```



git push origin --delete hotfix

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (hotfix)
$ git branch master
fatal: a branch named 'master' already exists

User@DESKTOP-372GENS MINGW64 ~/school-management-system (hotfix)
$ git checkout master
Switched to branch 'master'

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git commit -m "Исправен путь(почти)"
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    school-management-system/

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git commit -m "Исправен путь почти"
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git commit -m "Исправен путь(почти)"
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    school-management-system/

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git commit -m "Исправен путь почти"
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    school-management-system/

nothing added to commit but untracked files present (use "git add" to track)

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git branch -d hotfix
Deleted branch hotfix (was 417ea33).

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$
```



Такой подход помогает держать репозиторий аккуратным и организованным.

5. Чистка после нескольких веток

Если было создано несколько веток:

feature/schedule

И они все были успешно слиты в master, их можно удалить:

```
git branch -d feature/schedule
```

```
git branch -d feature/grades
```

```
git branch -d feature/diary
```

А также удалить на сервере:

```
git push origin --delete feature/schedule
```

```
ser@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git branch -d feature/schedule
error: branch 'feature/schedule' not found

ser@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
```

После этого история будет выглядеть чисто и понятно.

1. Просмотр списка всех веток

Чтобы увидеть все доступные ветки (включая удалённые):

```
git branch -a
```

Пример вывода:



* master

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git branch -a
* master
```

Таким образом, можно определить, какие ветки уже не нужны.

2. Проверка, какие ветки были слиты

Чтобы увидеть, какие ветки полностью слиты с master:

`git branch --merged master`

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git branch --merged master
* master

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$
```

Git покажет список веток, которые можно безопасно удалить:

feature/schedule

Это означает, что изменения из этой ветки уже добавлены в master.

3. Безопасное удаление ветки

Если ветка была слита, её можно удалить без потери данных:

`git branch -d feature/schedule`

Если попытаться удалить неслитую ветку, Git выдаст предупреждение:

error: The branch 'feature/diary' is not fully merged.

If you are sure you want to delete it, run 'git branch -D feature/diary'

4. Принудительное удаление неслитой ветки



Если ты уверен, что ветка больше не нужна, даже если она не была слита:

```
git branch -D feature/diary
```

Используется с осторожностью — такая команда не спрашивает подтверждения и удаляет ветку безвозвратно .

5. Удаление удалённой ветки на сервере

После слияния и локального удаления ветки, нужно удалить её и на GitHub/GitLab:

```
git push origin :feature/schedule
```

или более современная версия:

```
git push origin --delete feature/schedule
```

Так ты очищаешь историю на сервере.



Содержание

1. Инструктаж по соблюдению правил противопожарной безопасности, правил охраны труда, техники безопасности, санитарно-эпидемиологических правил и гигиенических нормативов
2. Ознакомление с инструментальными средствами
3. Сбор информации об объекте практики и анализ содержания источников
4. Экспериментально-практическая работа. Приобретение необходимых умений и первоначального практического опыта работы по специальности в рамках освоения вида деятельности ВД 2. Осуществление интеграции программных модулей



5. Обработка и систематизация полученного фактического материала

Организационный этап

Правила внутреннего распорядка, правила и нормы охраны труда, техники безопасности при работе с вычислительной техникой

Я, Старчик Артемий Филиппович, проходил(а) учебную практику в лабораторных условиях на базе Университета «Синергия».

При выполнении индивидуального задания по практике решал(а) кейс № 11 по интеграции Интеграция программных модулей в школьной системе.

Перед началом практики:

- Принял(а) участие в организационном собрании по практике.
- Ознакомил(а)сь с комплектом шаблонов отчетной документации по практике.
- Уточнил(а) контакты руководителя практики от Образовательной организации, а также правила в отношении субординации, внешнего вида, графика работы, техники безопасности:

Требования к внешнему виду: удобная одежда

График работы: с 13:00 по 15:00.(2 ч.); с 17:00 по 19:00.(2 ч.)

Круг обязанностей: прописание кода на git bush и предоставление понятного и читаемого отчета и кода



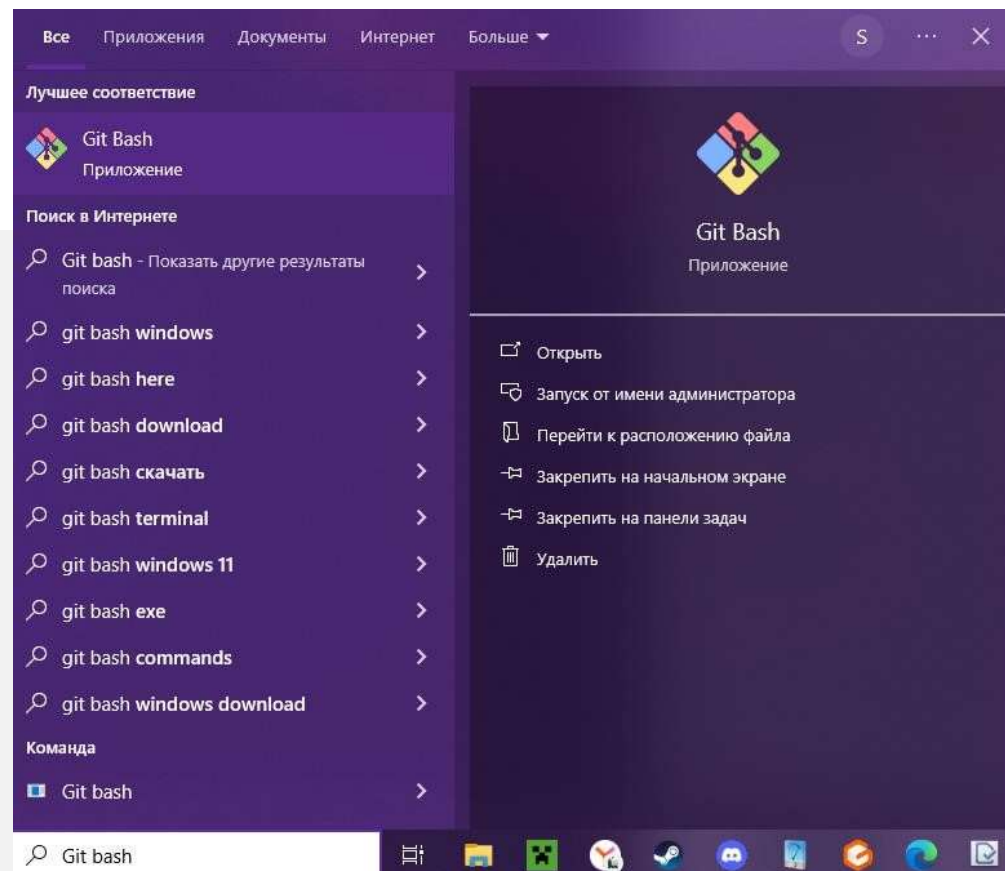
Доступ к данным: презентация и docx документы в приложении

Подготовительный этап

Ознакомление с ПО

Ознакомиться с инструментальными средствами для выполнения учебной практики и осуществить предустановку программного обеспечения.





Исследовательский этап

Сбор информации об объекте практики и анализ содержания источников

Школа: расписание уроков, ведение дневников, оценки, электронный журнал..





Этап проектирования

Использование системы контроля версий и методов для получения кода с заданной функциональностью и степенью качества

Скриншоты настройки пользователя, создания репозитория и коммита последнего изменения.

```
User@DESKTOP-372GEN5 MINGW64 ~  
$ git config --global user.name "MrZhr4ik"  
  
User@DESKTOP-372GEN5 MINGW64 ~  
$ git config --global user.email "starchik8116@gmail.com"
```

```
User@DESKTOP-372GEN5 MINGW64 ~/school-management-system  
$ git init  
Initialized empty Git repository in C:/Users/User/school-management-system/.git/
```

```
User@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)  
$ git commit -m "Initial commit"  
[master (root-commit) 9603f95] Initial commit  
2 files changed, 1 insertion(+)  
create mode 100644 README.md  
create mode 100644 hello.html
```



Этап проектирования

Использование системы контроля версий и методов для получения кода с заданной функциональностью и степенью качества

Скриншоты истории коммитов вашего проекта, алиасов и удаление тега.

```
MINGW64/c/Users/User/school-management-system
$ git log
commit f639bbde866f63c9d9dbbcae6734b243206cf4b6 (HEAD -> master)
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:47:56 2025 +0300

    Добавлено расписание на среду

commit 4bf2d4df09e9483b9ed2d06ec3384fcaa39de761
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:42:35 2025 +0300

    Добавлено расписание на вторник

commit d5af08f461d0828ad28ae76dd1e14f58047e8c59
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:36:26 2025 +0300

    Добавлено начальное расписание

commit 9603f95f0fa0b59cadd6c6fa66dc6582a17a763e
Author: MrZhr4ik <starchik8116@gmail.com>
Date: Thu Jun 5 13:09:42 2025 +0300

    Initial commit
```

```
ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
git log --all --pretty=format:"%h %cd | %s%d [Nan]" --graph --date=short
f639bbd 2025-06-05 | Добавлено расписание на среду (HEAD -> master) [MrZhr4ik]
4bf2d4d 2025-06-05 | Добавлено расписание на вторник [MrZhr4ik]
d5af08f 2025-06-05 | Добавлено начальное расписание [MrZhr4ik]
9603f95 2025-06-05 | Initial commit [MrZhr4ik]

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
```

```
MINGW64/c/Users/User/school-management-system
git config --global alias.ci commit

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
git config --global alias.ci commit

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
git config --global alias.st status

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
git config --global alias.br branch

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
git config --global alias.hist "log --pretty=format:'%h %ad | %s%d [Nan]' --graph --date=short"

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
git hist
f639bbd 2025-06-05 | Добавлено расписание на среду (HEAD -> master) [MrZhr4ik]
4bf2d4d 2025-06-05 | Добавлено расписание на вторник [MrZhr4ik]
d5af08f 2025-06-05 | Добавлено начальное расписание [MrZhr4ik]
9603f95 2025-06-05 | Initial commit [MrZhr4ik]

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
```



Этап проектирования

Использование системы контроля версий и методов для получения кода с заданной функциональностью и степенью качества

На данном слайде необходимо продемонстрировать скриншот файла конфигурации.

```
e1ena@LAPTOP-I3I3QM5I MINGW64 ~/Documents/hello (master)
$ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = false
    bare = false
    logallrefupdates = true
    symlinks = false
    ignorecase = true
```



Этап проектирования

Использование системы контроля версий и методов для получения кода с заданной функциональностью и степенью качества

Скриншоты создания ветки, просмотра истории ветки style, разрешения конфликтов.

```
ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
$ git checkout schedule.html
Updated 0 paths from the index

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
$ git reset HEAD schedule.html

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
$ git checkout schedule.html
Updated 0 paths from the index

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
```

```
User@DESKTOP-372GEN5 MINGW64 ~/school-management-system ((4bf2d4d...))
$ git checkout master
Previous HEAD position was 4bf2d4d Добавлено расписание на вторник
Switched to branch 'master'

User@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
```

```
ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
$ git log --all --pretty=format:"%h %cd | %s%d [%an]" --graph --date=short
f639bbd 2025-06-05 | Добавлено расписание на среду (HEAD -> master) [MrZhr4ik]

4bf2d4d 2025-06-05 | Добавлено расписание на вторник [MrZhr4ik]
d5af08f 2025-06-05 | Добавлено начальное расписание [MrZhr4ik]
9603f95 2025-06-05 | Initial commit [MrZhr4ik]

ser@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
```

Скриншот удаленных веток.



Этап проектирования

Использование системы контроля версий и методов для получения кода с заданной функциональностью и степенью качества

```
User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ git branch -a
* master

User@DESKTOP-372GENS MINGW64 ~/school-management-system (master)
$ |
```

Скриншот результата работы созданного алиаса.

```
eIena@LAPTOP-I3I3QM5I MINGW64 ~/Documents/hello (version2)
$ git last
commit c231127b40601b59292a5e3b0de3854e5341524c (HEAD -> version2, tag: v2.1, shared/master, master)
Author: ElenaVerstova <isip_e.v.verstova@mpt.ru>
Date:   Wed Mar 10 14:35:49 2021 +0300

    Added shared comment to README

eIena@LAPTOP-I3I3QM5I MINGW64 ~/Documents/hello (version2)
$ |
```

На данном слайде необходимо продемонстрировать скриншот отправки изменений в удаленный репозиторий. (У меня нету иконки регистрации на гит, уже зарегистрирован).



Этап проектирования

Использование системы контроля версий и методов для получения кода с заданной функциональностью и степенью качества

```
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 266 bytes | 266.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/MrZhr4ik/school-management-system/pull/new/maste
remote:
To https://github.com/MrZhr4ik/school-management-system
* [new branch]      master -> master

User@DESKTOP-372GEN5 MINGW64 ~/school-management-system (master)
$
```

На данном слайде необходимо продемонстрировать скриншот удаления ветки iss53.

```
eIena@LAPTOP-I3I3QM5I MINGW64 ~/Documents/hello_two (master)
$ git branch -d iss53
Deleted branch iss53 (was 97393ba).

eIena@LAPTOP-I3I3QM5I MINGW64 ~/Documents/hello_two (master)
$
```

На данном слайде необходимо продемонстрировать скриншот слияния веток.



Этап проектирования

Использование системы контроля версий и методов для получения кода с заданной функциональностью и степенью качества

```
elena@LAPTOP-I3I3QM5I MINGW64 ~/Documents/hello_two (se)
$ git checkout master
Switched to branch 'master'

elena@LAPTOP-I3I3QM5I MINGW64 ~/Documents/hello_two (master)
$ git merge se
Updating 08d256f..4a98db2
Fast-forward
 e.txt | 0
 q.txt | 0
 repo1 | 1 +
3 files changed, 1 insertion(+)
create mode 100644 e.txt
create mode 100644 q.txt
create mode 160000 repo1
```



Отчетный этап

Формирование отчетной документации по результатам работ

При оформлении отчетных материалов следует придерживаться действующих стандартов.

- В соответствии с ГОСТ 2.105-79 «Общие требования к текстовым документам» иллюстрации (графики, схемы, диаграммы) могут быть приведены как в основном тексте, так и в приложении. Все иллюстрации именуют рисунками. Все рисунки, таблицы и формулы нумеруют арабскими цифрами последовательно (сквозная нумерация) или в пределах раздела (относительная нумерация). В приложении - в пределах приложения. Каждый рисунок должен иметь подрисуночную подпись - название, помещаемую под рисунком.
- Рисунки следует размещать так, чтобы их можно было рассматривать без поворота страницы. Если такое размещение невозможно, рисунки следует располагать так, чтобы для просмотра надо было повернуть страницу по часовой стрелке. В этом случае верхним краем является левый край страницы. Расположение и размеры полей сохраняются.



Отчетный этап

- Номер таблицы размещают в правом верхнем углу или перед заголовком таблицы, если он есть. Заголовок, кроме первой буквы, выполняют строчными буквами. Ссылки на таблицы в тексте пояснительной записки указывают в виде слова «табл.» и номера таблицы. *Например:*

Результаты тестов приведены в табл. 4.

Формирование отчетной документации по результатам работ

При оформлении отчетных материалов следует придерживаться действующих стандартов.

- Список литературы должен включать все использованные источники. Сведения о книгах (монографиях, учебниках, пособиях, справочниках и т.д.) должны содержать: фамилию и инициалы автора, заглавие книги, место издания, издательство, год издания. При наличии трех и более авторов допускается указывать фамилию и инициалы только первого из них со словами «и др.». Издательство надо приводить полностью в именительном падеже: допускается сокращение названия только двух городов: Москва (М.) и Санкт-Петербург (СПб.).



Отчетный этап

- Сведения о статье из периодического издания должны включать: фамилию и инициалы автора, наименование статьи, издания (журнала), серии (если она есть), год выпуска, том (если есть), номер издания (журнала) и номера страниц, на которых помещена статья.
- При ссылке на источник из списка литературы (особенно при обзоре аналогов) надо указывать порядковый номер по списку литературы, заключенный в квадратные скобки; например: [5].

Выводы о результатах прохождения учебной практики:

выполняемая работа, приобретенные умения и навыки

Подведите итоги прохождения учебной практики:

В ходе прохождения учебной практики мной были получены освоены следующие навыки:

1. Работа с Git: ветки, коммиты, слияние, перебазирование



Отчетный этап

2. Создание HTML/CSS-проекта: расписание, оценки, дневник, журнал
3. Навигация по истории: алиасы, фильтры, просмотр изменений
4. Разрешение конфликтов и управление версиями
5. Работа с GitHub: клонирование, отправка изменений, теги, удалённые ветки



Отчетный этап

Список используемой литературы

1. Советов, Б. Я. Базы данных: учебник для СПО / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 3-е изд., перераб. и доп. — Москва: Издательство Юрайт, 2023. — 420 с. — (Профессиональное образование). — ISBN 978-5-534-09324-7. — Текст: электронный // Образовательная платформа Юрайт. — URL: <https://urait.ru/bcode/514585>
2. Стружкин, Н. П. Базы данных: проектирование: учебник для СПО / Н. П. Стружкин, В. В. Годин. — Москва: Издательство Юрайт, 2023. — 477 с. — (Профессиональное образование). — ISBN 978-5-534-11635-9. — Текст: электронный // Образовательная платформа Юрайт. — URL: <https://urait.ru/bcode/518499>
3. Нагаева, И. А. Основы алгоритмизации и программирования: практикум / И. А. Нагаева, И. А. Кузнецов. — Москва; Берлин: Директ-Медиа, 2021. — 169 с. — Режим доступа: по подписке. — URL: <https://biblioclub.ru>
4. Приложения 1.1–1.9 из файла Git.docx — методические указания по работе с Git
5. Git Book (рус.) — официальное руководство по Git
6. URL: <https://git-scm.com/book/ru/v2>
7. learngitbranching.js.org — интерактивное обучение Git



8. URL: <https://learngitbranching.js.org>
9. GitHub — репозиторий проекта
10. URL: <https://github.com/MrZhr4ik/school-management-system>
11. Шаблон отчетной презентации (PDF)

Отчетный этап

Приложения

[1.1. Гит.docx](#)

[1.2. Гит.docx](#)

[1.3. Гит.docx](#)

[1.4. Гит.docx](#)

[1.5. Гит.docx](#)

[1.6. Гит.docx](#)



[1.7. Гит.docx](#)

[1.8. Гит.docx](#)

[1.9. Гит.docx](#)

