

Лабораторная работа №4

Продвинутое использование git

Mohamed Musa

2025-10-13

Содержание I

1. Информация
2. Вводная часть
3. Теоретические сведения
4. Выполнение работы
5. Результаты
6. Заключение

Раздел 1

1. Информация

1.1 Докладчик

► Mohamed Musa

1.1 Докладчик

- ▶ Mohamed Musa
- ▶ Студент группы НКАбд-05-24

1.1 Докладчик

- ▶ Mohamed Musa
- ▶ Студент группы НКАбд-05-24
- ▶ Студенческий билет: 1032248286

1.1 Докладчик

- ▶ Mohamed Musa
- ▶ Студент группы НКАбд-05-24
- ▶ Студенческий билет: 1032248286
- ▶ Российский университет дружбы народов

1.1 Докладчик

- ▶ Mohamed Musa
- ▶ Студент группы НКАбд-05-24
- ▶ Студенческий билет: 1032248286
- ▶ Российский университет дружбы народов
- ▶ 1032248286@pfur.ru

Раздел 2

2. Вводная часть

2.1 Актуальность

- ▶ Правильная организация работы с системой контроля версий критически важна для командной разработки

2.1 Актуальность

- ▶ Правильная организация работы с системой контроля версий критически важна для командной разработки
- ▶ Git-flow предоставляет структурированный подход к ветвлению

2.1 Актуальность

- ▶ Правильная организация работы с системой контроля версий критически важна для командной разработки
- ▶ Git-flow предоставляет структурированный подход к ветвлению
- ▶ Семантическое версионирование обеспечивает понятную систему нумерации релизов

2.1 Актуальность

- ▶ Правильная организация работы с системой контроля версий критически важна для командной разработки
- ▶ Git-flow предоставляет структурированный подход к ветвлению
- ▶ Семантическое версионирование обеспечивает понятную систему нумерации релизов
- ▶ Conventional commits стандартизируют историю изменений

2.2 Объект и предмет исследования

► Система контроля версий Git

2.2 Объект и предмет исследования

- ▶ Система контроля версий Git
- ▶ Модель ветвления Gitflow

2.2 Объект и предмет исследования

- ▶ Система контроля версий Git
- ▶ Модель ветвления Gitflow
- ▶ Инструменты автоматизации работы с репозиториями

2.2 Объект и предмет исследования

- ▶ Система контроля версий Git
- ▶ Модель ветвления Gitflow
- ▶ Инструменты автоматизации работы с репозиториями
- ▶ Стандарты оформления коммитов и версионирования

2.3 Цели и задачи

- ▶ Получить навыки работы с git-flow

2.3 Цели и задачи

- ▶ Получить навыки работы с git-flow
- ▶ Освоить семантическое версионирование

2.3 Цели и задачи

- ▶ Получить навыки работы с git-flow
- ▶ Освоить семантическое версионирование
- ▶ Научиться использовать conventional commits

2.3 Цели и задачи

- ▶ Получить навыки работы с git-flow
- ▶ Освоить семантическое версионирование
- ▶ Научиться использовать conventional commits
- ▶ Автоматизировать создание changelog

2.4 Материалы и методы

▶ Git и расширение git-flow

2.4 Материалы и методы

- ▶ Git и расширение git-flow
- ▶ Node.js и пакетный менеджер npm

2.4 Материалы и методы

- ▶ Git и расширение git-flow
- ▶ Node.js и пакетный менеджер npm
- ▶ Инструменты: commitizen, standard-changelog

2.4 Материалы и методы

- ▶ Git и расширение git-flow
- ▶ Node.js и пакетный менеджер npm
- ▶ Инструменты: commitizen, standard-changelog
- ▶ Тестовый репозиторий git-extended

Раздел 3

3. Теоретические сведения

3.1 Gitflow Workflow

- ▶ Модель ветвления, опубликованная Винсентом Дриссенем

3.1 Gitflow Workflow

- ▶ Модель ветвления, опубликованная Винсентом Дриссенем
- ▶ Строгая модель с учётом выпуска проекта

3.1 Gitflow Workflow

- ▶ Модель ветвления, опубликованная Винсентом Дриссенем
- ▶ Строгая модель с учётом выпуска проекта
- ▶ Основные ветки: **master** и **develop**

3.1 Gitflow Workflow

- ▶ Модель ветвления, опубликованная Винсентом Дриссенем
- ▶ Строгая модель с учётом выпуска проекта
- ▶ Основные ветки: **master** и **develop**
- ▶ Вспомогательные ветки: **feature**, **release**, **hotfix**

3.2 Схема работы Gitflow

► Из master создаётся develop

3.2 Схема работы Gitflow

- ▶ Из master создаётся develop
- ▶ Из develop создаются feature ветки

3.2 Схема работы Gitflow

- ▶ Из master создаётся develop
- ▶ Из develop создаются feature ветки
- ▶ Feature ветки сливаются обратно в develop

3.2 Схема работы Gitflow

- ▶ Из master создаётся develop
- ▶ Из develop создаются feature ветки
- ▶ Feature ветки сливаются обратно в develop
- ▶ Из develop создаются release ветки

3.2 Схема работы Gitflow

- ▶ Из master создаётся develop
- ▶ Из develop создаются feature ветки
- ▶ Feature ветки сливаются обратно в develop
- ▶ Из develop создаются release ветки
- ▶ Release сливается в master и develop

3.2 Схема работы Gitflow

- ▶ Из master создаётся develop
- ▶ Из develop создаются feature ветки
- ▶ Feature ветки сливаются обратно в develop
- ▶ Из develop создаются release ветки
- ▶ Release сливается в master и develop
- ▶ Hotfix создаётся из master и сливается в обе ветки

3.3 Семантическое версионирование

Формат версии: **МАЖОРНАЯ.МИНОРНАЯ.ПАТЧ**

► **МАЖОРНАЯ** — обратно несовместимые изменения API

Пример: 1 . 2 . 3 → 2 . 0 . 0 (breaking change)

3.3 Семантическое версионирование

Формат версии: **МАЖОРНАЯ.МИНОРНАЯ.ПАТЧ**

- ▶ **МАЖОРНАЯ** — обратно несовместимые изменения API
- ▶ **МИНОРНАЯ** — новая функциональность с обратной совместимостью

Пример: 1 . 2 . 3 → 2 . 0 . 0 (breaking change)

3.3 Семантическое версионирование

Формат версии: **МАЖОРНАЯ.МИНОРНАЯ.ПАТЧ**

- ▶ **МАЖОРНАЯ** — обратно несовместимые изменения API
- ▶ **МИНОРНАЯ** — новая функциональность с обратной совместимостью
- ▶ **ПАТЧ** — обратно совместимые исправления ошибок

Пример: 1 . 2 . 3 → 2 . 0 . 0 (breaking change)

3.4 Conventional Commits

Структура коммита:

<[4][char]> (<[7][char]>) : <[71][char]>

[4][char]

[7][char] [71][char]

3.5 Типы коммитов

► **feat:** — новая функция (MINOR)

3.5 Типы коммитов

- ▶ **feat:** — новая функция (MINOR)
- ▶ **fix:** — исправление ошибки (PATCH)

3.5 Типы коммитов

- ▶ **feat:** — новая функция (MINOR)
- ▶ **fix:** — исправление ошибки (PATCH)
- ▶ **docs:** — изменения в документации

3.5 Типы коммитов

- ▶ **feat:** — новая функция (MINOR)
- ▶ **fix:** — исправление ошибки (PATCH)
- ▶ **docs:** — изменения в документации
- ▶ **style:** — форматирование кода

3.5 Типы коммитов

- ▶ **feat:** — новая функция (MINOR)
- ▶ **fix:** — исправление ошибки (PATCH)
- ▶ **docs:** — изменения в документации
- ▶ **style:** — форматирование кода
- ▶ **refactor:** — рефакторинг

3.5 Типы коммитов

- ▶ **feat:** — новая функция (MINOR)
- ▶ **fix:** — исправление ошибки (PATCH)
- ▶ **docs:** — изменения в документации
- ▶ **style:** — форматирование кода
- ▶ **refactor:** — рефакторинг
- ▶ **test:** — добавление тестов

3.5 Типы коммитов

- ▶ **feat:** — новая функция (MINOR)
- ▶ **fix:** — исправление ошибки (PATCH)
- ▶ **docs:** — изменения в документации
- ▶ **style:** — форматирование кода
- ▶ **refactor:** — рефакторинг
- ▶ **test:** — добавление тестов
- ▶ **chore:** — изменения в сборке

3.6 Инструменты

► **git-flow** — расширение Git для Gitflow

3.6 Инструменты

- ▶ **git-flow** — расширение Git для Gitflow
- ▶ **pnpm** — быстрый пакетный менеджер

3.6 Инструменты

- ▶ **git-flow** — расширение Git для Gitflow
- ▶ **pnpm** — быстрый пакетный менеджер
- ▶ **commitizen** — помощник для создания коммитов

3.6 Инструменты

- ▶ **git-flow** — расширение Git для Gitflow
- ▶ **pnpm** — быстрый пакетный менеджер
- ▶ **commitizen** — помощник для создания коммитов
- ▶ **standard-changelog** — генератор changelog

Раздел 4

4. Выполнение работы

4.1 Установка программного обеспечения

Установлены следующие компоненты:

- ▶ git-flow (из репозитория Copr)

4.1 Установка программного обеспечения

Установлены следующие компоненты:

- ▶ git-flow (из репозитория Copr)
- ▶ Node.js и npm

4.1 Установка программного обеспечения

Установлены следующие компоненты:

- ▶ git-flow (из репозитория Copr)
- ▶ Node.js и npm
- ▶ commitizen (глобально через npm)

4.1 Установка программного обеспечения

Установлены следующие компоненты:

- ▶ git-flow (из репозитория Copr)
- ▶ Node.js и npm
- ▶ commitizen (глобально через npm)
- ▶ standard-changelog (глобально через npm)

4.2 Установка git-flow

Для Fedora:

```
dnf copr enable elegos/gitflow  
dnf install gitflow
```

4.3 Установка Node.js

Установка необходимых пакетов:

```
dnf install nodejs  
dnf install pnpm
```

4.4 Настройка pnpm

Выполнена настройка pnpm package manager:

```
pnpm setup  
source ~/.bashrc
```

4.5 Настройка rnpm (скриншот)

```
[ceazer@fedora git-extended]$ nano package.json
[ceazer@fedora git-extended]$ git add .
[ceazer@fedora git-extended]$ git cz
Parsing JSON at /home/ceazer/work/git-extended/package.json for commitizen config failed:

[ceazer@fedora git-extended]$ nano package.json
[ceazer@fedora git-extended]$ git cz
Parsing JSON at /home/ceazer/work/git-extended/package.json for commitizen config failed:

[ceazer@fedora git-extended]$ git add .
[ceazer@fedora git-extended]$ git cz
Parsing JSON at /home/ceazer/work/git-extended/package.json for commitizen config failed:

[ceazer@fedora git-extended]$ nano package.json
[ceazer@fedora git-extended]$ git add .
[ceazer@fedora git-extended]$ git cz
Parsing JSON at /home/ceazer/work/git-extended/package.json for commitizen config failed:

[ceazer@fedora git-extended]$ git push
^C
[ceazer@fedora git-extended]$ git remote set-url git-extended git@github.com:cezaryt5/git-extended.git
error: Her rakoro smashero penoswropw wgit-extended
[ceazer@fedora git-extended]$ git remote -v
origin  git@github.com:cezaryt5/git-extended.git (fetch)
origin  git@github.com:cezaryt5/git-extended.git (push)
[ceazer@fedora git-extended]$ cat package.json
{
  "name": "git-extended",
  "version": "1.0.0",
  "description": "Git repo for educational purposes",
  "main": "index.js",
  "repository": "git@github.com:cezaryt5/git-extended.git",
  "author": "Mohammed Musa 1032248286@pfur.ru",
  "license": "CC-BY-4.0",
  "packageManager": "rpm@10.9.0"
  "config": {
    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}
```

4.6 Создание репозитория

Выполнены следующие шаги:

1. Создан репозиторий git-extended на GitHub

4.6 Создание репозитория

Выполнены следующие шаги:

1. Создан репозиторий git-extended на GitHub
2. Инициализирован локальный репозиторий

4.6 Создание репозитория

Выполнены следующие шаги:

1. Создан репозиторий git-extended на GitHub
2. Инициализирован локальный репозиторий
3. Выполнен первый коммит

4.6 Создание репозитория

Выполнены следующие шаги:

1. Создан репозиторий git-extended на GitHub
2. Инициализирован локальный репозиторий
3. Выполнен первый коммит
4. Настроено подключение к удаленному репозиторию

4.7 Конфигурация Node.js

Инициализация проекта:

```
pnpm init
```

Параметры:

► Название: git-extended

4.7 Конфигурация Node.js

Инициализация проекта:

```
pnpm init
```

Параметры:

- ▶ Название: git-extended
- ▶ Версия: 1.0.0

4.7 Конфигурация Node.js

Инициализация проекта:

```
pnpm init
```

Параметры:

- ▶ Название: git-extended
- ▶ Версия: 1.0.0
- ▶ Лицензия: CC-BY-4.0

4.7 Конфигурация Node.js

Инициализация проекта:

```
pnpm init
```

Параметры:

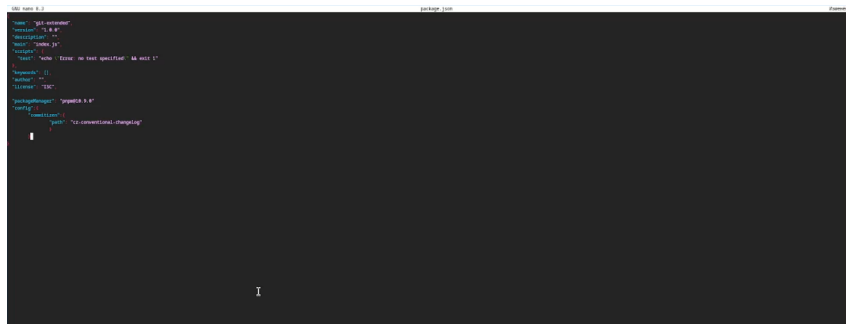
- ▶ Название: git-extended
- ▶ Версия: 1.0.0
- ▶ Лицензия: CC-BY-4.0
- ▶ Автор: Mohamed Musa

4.8 Файл package.json

Добавлена конфигурация для commitizen:

```
"config": {  
  "commitizen": {  
    "path": "cz-conventional-changelog"  
  }  
}
```

4.9 Файл package.json (скриншот)



```
082 xeno 8.2 package.json @xeno
{
  "name": "git-extensions",
  "version": "1.8.8",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo 'Error: no test specified.' && exit 1"
  },
  "dependencies": {},
  "author": "",
  "license": "ISC",
  "packageManager": "pnpm@8.6",
  "config": {
    "conventional": {
      "path": "cz-conventional-changelog"
    }
  }
}
```

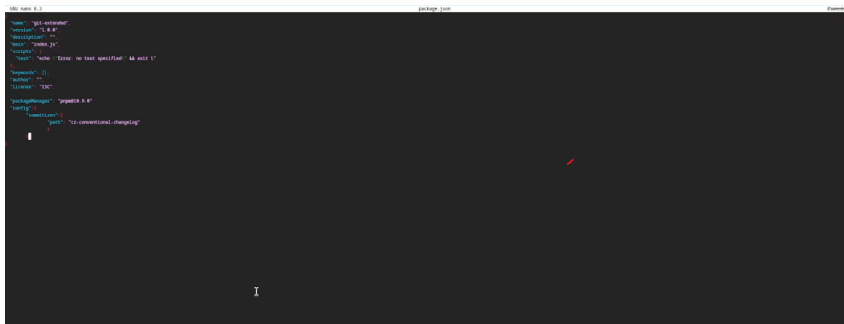
Рисунок 2: Конфигурация package.json

4.10 Установка commitizen

Установка инструментов для conventional commits:

```
pnpm add -g commitizen  
pnpm add -g standard-changelog
```

4.11 Установка commitizen (скриншот)



```
package.json
{
  "name": "git-extensions",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo 'Error: no test specified' && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "packageManager": "pnpm@8.6",
  "commitizen": {
    "path": "cz-conventional-changelog"
  }
}
```

Рисунок 3: Установка commitizen

4.12 Инициализация git-flow

Команды инициализации:

```
git flow init
```

Параметры:

- ▶ Префикс для тегов: v

4.12 Инициализация git-flow

Команды инициализации:

```
git flow init
```

Параметры:

- ▶ Префикс для тегов: v
- ▶ Ветка production: master

4.12 Инициализация git-flow

Команды инициализации:

```
git flow init
```

Параметры:

- ▶ Префикс для тегов: v
- ▶ Ветка production: master
- ▶ Ветка разработки: develop

4.13 Загрузка веток на GitHub

```
git push --all  
git branch --set-upstream-to=origin/develop develop
```

4.14 Создание первого релиза

Команды для создания релиза 1.0.0:

```
git flow release start 1.0.0
standard-changelog --first-release
git add CHANGELOG.md
git commit -am 'chore(site): add changelog'
git flow release finish 1.0.0
```

4.15 Создание первого релиза (скриншот)

```

Вспомогательск, что у нас есть необходимая права доступа
в персональном репозитории:
(ccazov@fedora git-extended)$ git push --all
^C
(ccazov@fedora git-extended)$ git push
fatal: The current branch develop has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin develop

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
(ccazov@fedora git-extended)$ git push --set-upstream origin develop
^C
(ccazov@fedora git-extended)$ git push --all
^C
(ccazov@fedora git-extended)$ git add package.json
(ccazov@fedora git-extended)$ git cz
No files added to staging! Did you forget to run git add?
(ccazov@fedora git-extended)$ git push
fatal: The current branch develop has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin develop

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
(ccazov@fedora git-extended)$ git push --all
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/ccazov5/git-extended/pull/new/develop
remote:
To github.com:ccazov5/git-extended git
+ (new branch)   develop -> develop
(ccazov@fedora git-extended)$ git branch --set-upstream-to=origin/develop develop
fatal: nexus develop не существует
(ccazov@fedora git-extended)$ git branch --set-upstream-to=origin/develop develop
branch 'develop' set up to track 'origin/develop'.
(ccazov@fedora git-extended)$ git branch --set-upstream-to=origin/develop develop
branch 'develop' set up to track 'origin/develop'.
(ccazov@fedora git-extended)$ git flow release start 1.0.0
Нажмите enter, чтобы начать новый релиз 1.0.0

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now
- Start committing last-minute files in preparing your release
- When done, run:

    git flow release finish 1.0.0

(ccazov@fedora git-extended)$ standard-changelog --first-release
(ccazov@fedora git-extended)$ git commit -m "chore(site): add changelog"
[release/1.0.0 36db745] chore(site): add changelog
1 file changed, 3 insertions(+), 16 deletions(-)
(ccazov@fedora git-extended)$

```

4.16 Публикация релиза

Отправка на GitHub:

```
git push --all  
git push --tags  
gh release create v1.0.0 -F CHANGELOG.md
```

4.17 Работа с changelog

Автоматическая генерация журнала изменений:

```
standard-changelog
```

Журнал содержит:

- ▶ Список новых функций (feat)

4.17 Работа с changelog

Автоматическая генерация журнала изменений:

```
standard-changelog
```

Журнал содержит:

- ▶ Список новых функций (feat)
- ▶ Исправления ошибок (fix)

4.17 Работа с changelog

Автоматическая генерация журнала изменений:

`standard-changelog`

Журнал содержит:

- ▶ Список новых функций (feat)
- ▶ Исправления ошибок (fix)
- ▶ Другие изменения

4.18 Работа с changelog (скриншот)

```
(cease@fedora git-extended)$ npm add -g standard-changelog
npm WARN deprecated standard-changelog@v0.0.0: git007 2.3, inflight0.0.0
Already up to date
Progress: evaluated 185, reused 185, downloaded 0, added 0, done
Done in 5.5s using npm v10.9.0
(cease@fedora git-extended)$ git push -u origin master
^C
(cease@fedora git-extended)$ git add .
(cease@fedora git-extended)$ git cz
Putting /ZOM at /home/cease/work/git-extended/package.json for commitizen config failed:

(cease@fedora git-extended)$ cat package.json | grep -v '"4"' | sed 's/././g'
{
  "name": "git-extended",
  "version": "1.0.0",
  "description": "Git repo for educational purposes",
  "main": "index.js",
  "repository": "git@github.com:ceasey85/git-extended.git",
  "author": "Mohamed Muta 183248260@fud.zu",
  "license": "CC-BY-4.0",
  "packageManager": "pnpm@8.9.0",
  "config": {
    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}

(cease@fedora git-extended)$ npm install jq
Данное действие выполняется с правами root. Продолжить, войдя в систему как пользователь с повышенными правами или используйте опции "--assume" или "--downloadonly", чтобы выполнить команду без изменения состояния системы.
(cease@fedora git-extended)$ sudo npm install jq
[sudo] пароль для cease:
Обновление и установка пакетов завершено.
Репозитории загрузены.
Пакет "jq-1.7.1-11.fc42.x86_64" уже установлен.

Ничего не делать.
(cease@fedora git-extended)$ jq . package.json
jq: parse error: Expected separator between values at line 10, column 10
(cease@fedora git-extended)$ nano package.json
(cease@fedora git-extended)$ git add .
(cease@fedora git-extended)$ git cz
cz:cli@4.3.1, cz-conventional-changelog@3.3.0

Select the type of change that you're committing: feat: A new feature
What is the scope of this change (e.g. component or file name): (press enter to skip) package.json
Write a short, imperative tense description of the change (max 80 chars):
(1) my first cz git
Provide a longer description of the change: (press enter to skip)

Are there any breaking changes? no
A BREAKING CHANGE commit requires a body. Please enter a longer description of the commit itself:
(2)
Describe the breaking changes:

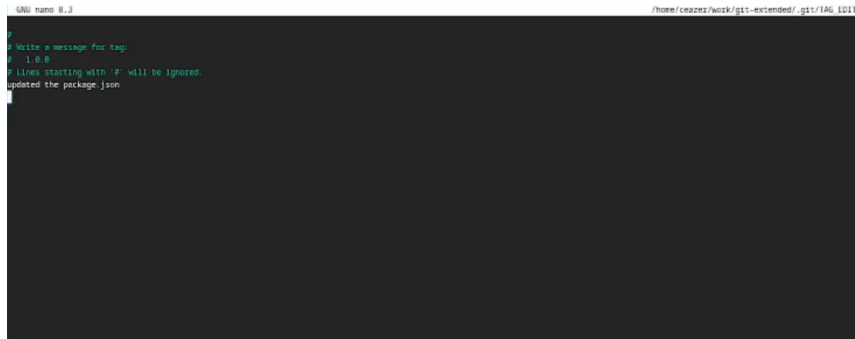
Does this change affect any open issues? no
(master 57ffac1) feat(package.json): my first cz git
1 file changed, 12 insertions(+), 13 deletions(-)
(cease@fedora git-extended)$
```

4.19 Добавление записей в changelog

Процесс обновления changelog при новом релизе:

```
git add CHANGELOG.md  
git commit -am 'chore(site): update changelog'
```

4.20 Добавление записей в changelog (скриншот)




```
GNU nano 8.3 /home/ceazer/work/git-extended/.git/1A6.ID11
#
# Write a message for tag:
# 1.0.0
# Lines starting with '#' will be ignored.
updated the package.json
```

Рисунок 6: Обновление changelog

4.21 Разработка новой функциональности

Работа с feature ветками:

```
git flow feature start feature_branch  
# ...  ...  
git flow feature finish feature_branch
```

4.22 Создание нового релиза

Процесс создания релиза 1.2.3:

1. Создание ветки релиза

4.22 Создание нового релиза

Процесс создания релиза 1.2.3:

1. Создание ветки релиза
2. Обновление версии в `package.json`

4.22 Создание нового релиза

Процесс создания релиза 1.2.3:

1. Создание ветки релиза
2. Обновление версии в `package.json`
3. Генерация changelog

4.22 Создание нового релиза

Процесс создания релиза 1.2.3:

1. Создание ветки релиза
2. Обновление версии в `package.json`
3. Генерация changelog
4. Завершение релиза

4.22 Создание нового релиза

Процесс создания релиза 1.2.3:

1. Создание ветки релиза
2. Обновление версии в `package.json`
3. Генерация changelog
4. Завершение релиза
5. Отправка на GitHub

4.22 Создание нового релиза

Процесс создания релиза 1.2.3:

1. Создание ветки релиза
2. Обновление версии в `package.json`
3. Генерация changelog
4. Завершение релиза
5. Отправка на GitHub
6. Создание релиза на GitHub

Раздел 5

5. Результаты

5.1 Полученные навыки

- ▶ Работа с моделью ветвления Gitflow

5.1 Полученные навыки

- ▶ Работа с моделью ветвления Gitflow
- ▶ Применение семантического версионирования

5.1 Полученные навыки

- ▶ Работа с моделью ветвления Gitflow
- ▶ Применение семантического версионирования
- ▶ Создание стандартизированных коммитов

5.1 Полученные навыки

- ▶ Работа с моделью ветвления Gitflow
- ▶ Применение семантического версионирования
- ▶ Создание стандартизированных коммитов
- ▶ Автоматическая генерация документации

5.1 Полученные навыки

- ▶ Работа с моделью ветвления Gitflow
- ▶ Применение семантического версионирования
- ▶ Создание стандартизированных коммитов
- ▶ Автоматическая генерация документации
- ▶ Профессиональная организация рабочего процесса

Раздел 6

6. Заключение

6.1 Выводы

Освоены современные практики работы с Git: