МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт № 3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Программное обеспечение средств вычислительной техники и автоматизированных систем»

**Отчёт по практической работе №2**

**по курсу «Технологии разработки программного обеспечения»**

**по теме: «CI/CD»**

**Вариант 1**

Выполнил:

Благушин Н.А.

Группа: М3О-107СВ-24

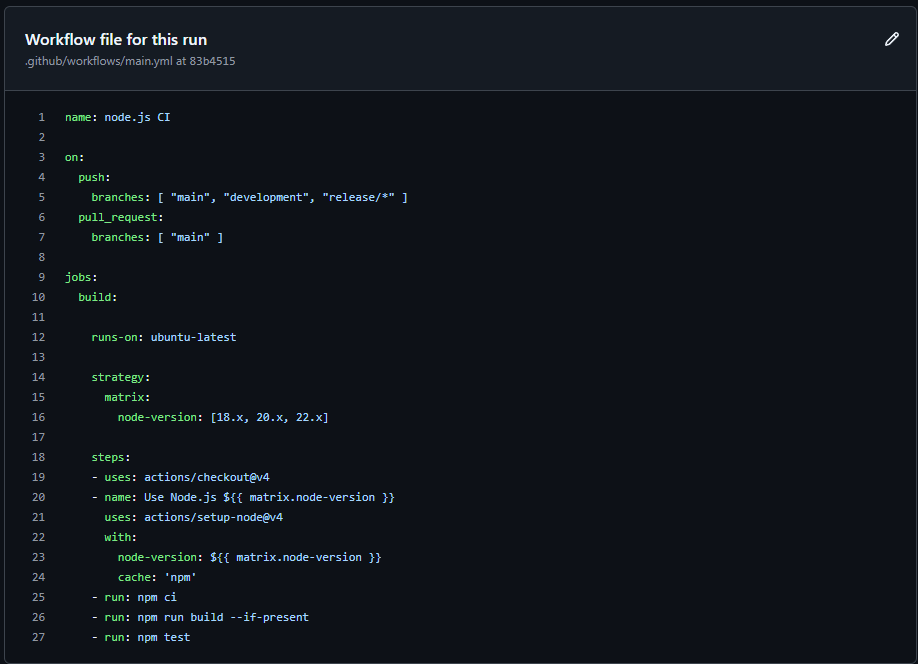
Приняли:

Титов Юрий Павлович

Москва 2024 г.

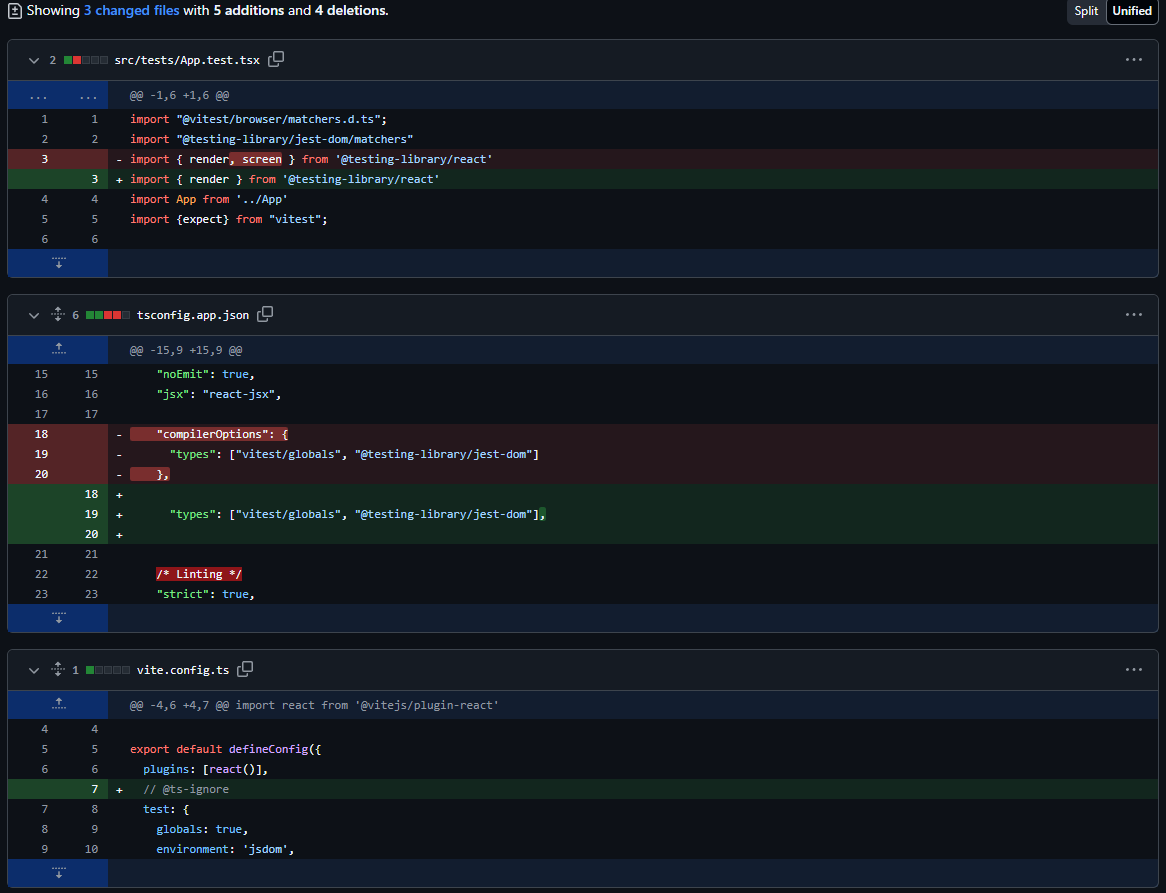
1. Настройте систему непрерывной интеграции (CI) для вашего проекта, например, используя сервисы GitHub Actions, GitLab CI/CD, Jenkins, Travis CI или другие инструменты. Пример «Настройка CI с помощью GitHub Actions»:
   1. Откройте репозиторий проекта на GitHub. Перейдите во вкладку "Actions" в верхней части страницы репозитория. Нажмите на кнопку "Set up a workflow yourself" или выберите один из шаблонов для создания нового workflow.
   2. Создайте файл с названием main.yml в папке .github/workflows/ вашего репозитория и определите в нем шаги вашего CI/CD процесса.
   3. Сделайте коммит и отправьте изменения в ваш репозиторий на GitHub.
   4. Проверка работы CI: После отправки изменений в репозиторий, перейдите во вкладку "Actions" на GitHub. Отобразите в отчете список запущенных workflow и их статус. Проверьте логи выполнения шагов вашего CI/CD процесса.
2. Настройте CI таким образом, чтобы он автоматически запускался при каждом новом коммите в ваш репозиторий.

Настроена система CI через GitHub Actions с помощью готового шаблона для node.js (в нашем случае проект на vite react ts, поэтому необходимо использовать node.js)

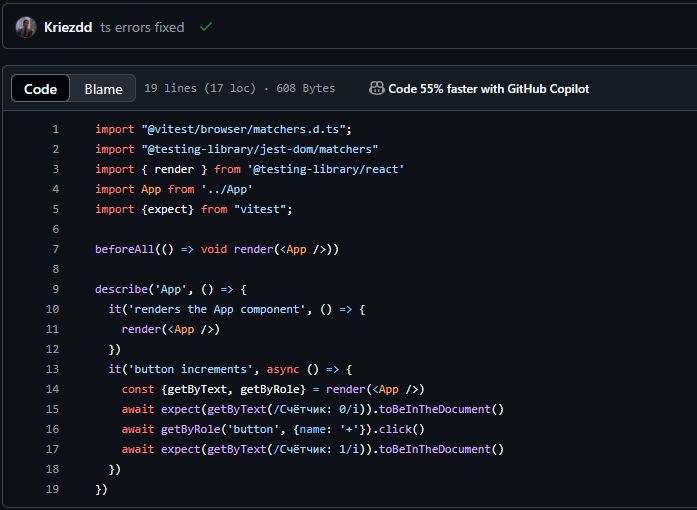


1. Добавьте скрипты для тестирования вашего проекта (например, unit-тесты, интеграционные тесты) и убедитесь, что они успешно проходят при запуске через CI.
2. Создайте ветку development для разработки новых функций и исправлений.
3. Разработайте новую функцию или исправление ошибки в вашем проекте и закоммитьте изменения в ветку development.
4. Убедитесь, что CI успешно запустился после коммита в ветку development и прошел все тесты.

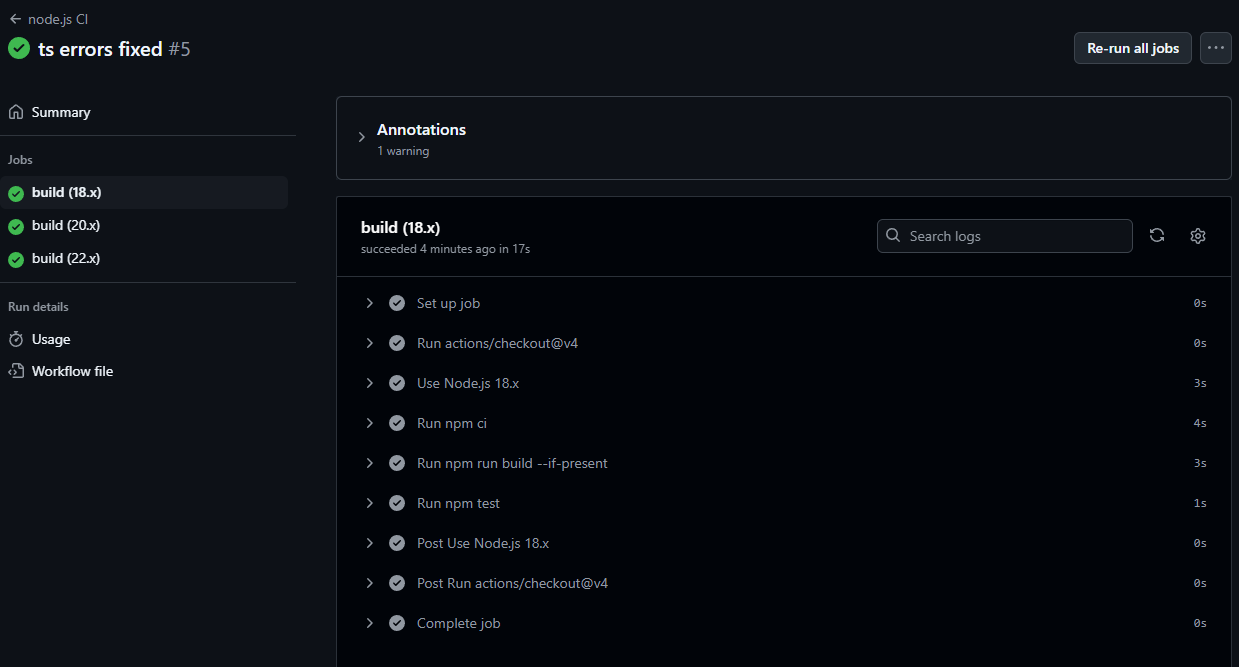
Коммит в ветке development c исправлением ошибок TypeScript:

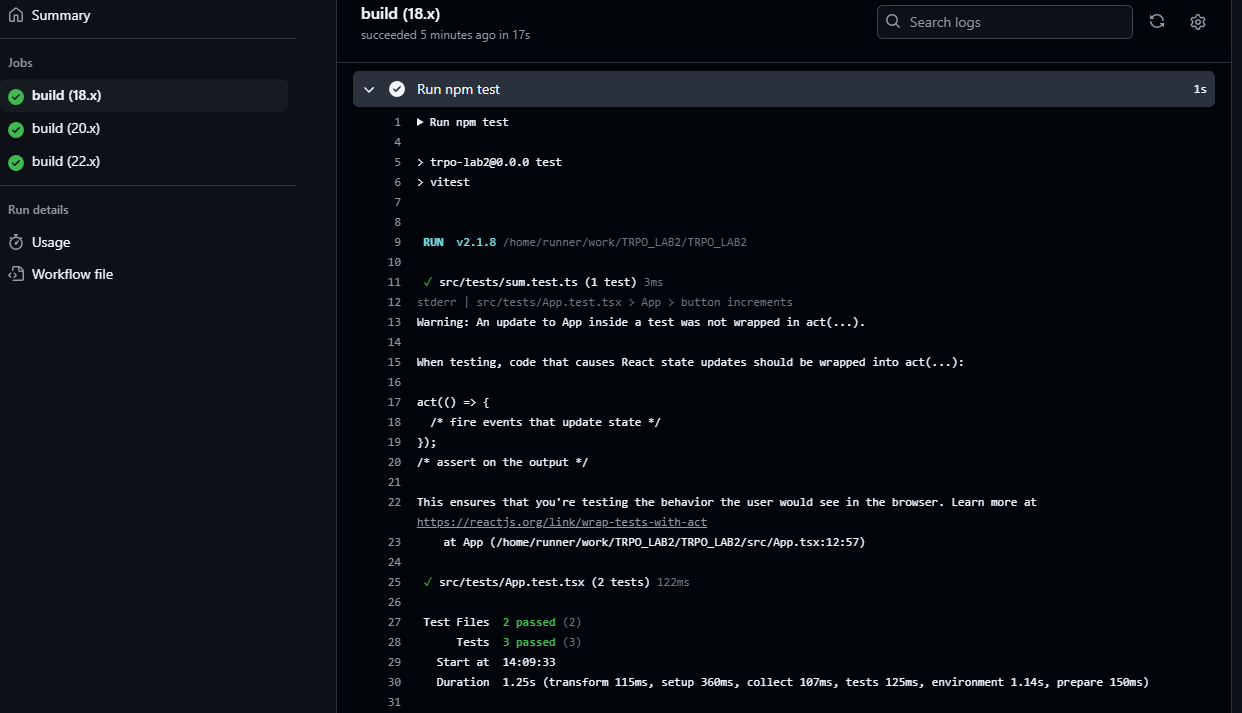


Скрипт тестов для проверки рендера компонента и правильной работы кнопки на странице:

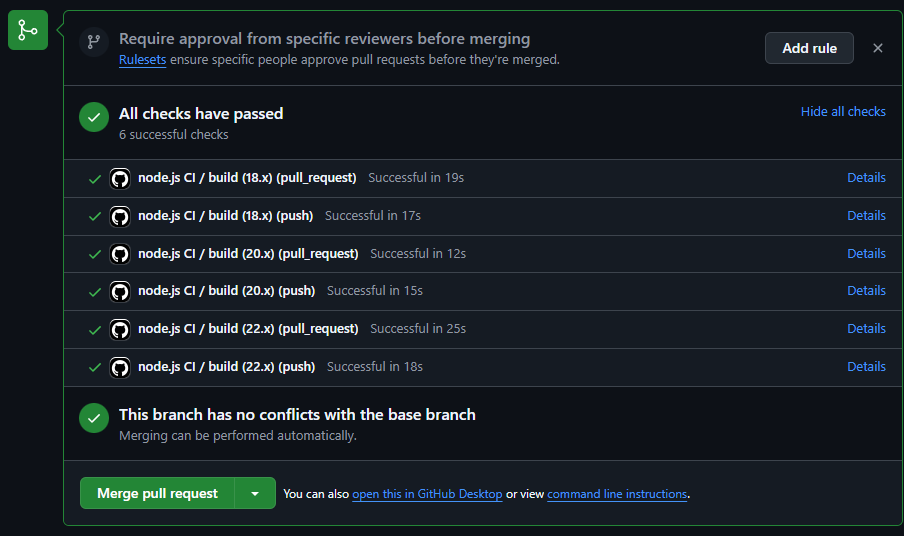


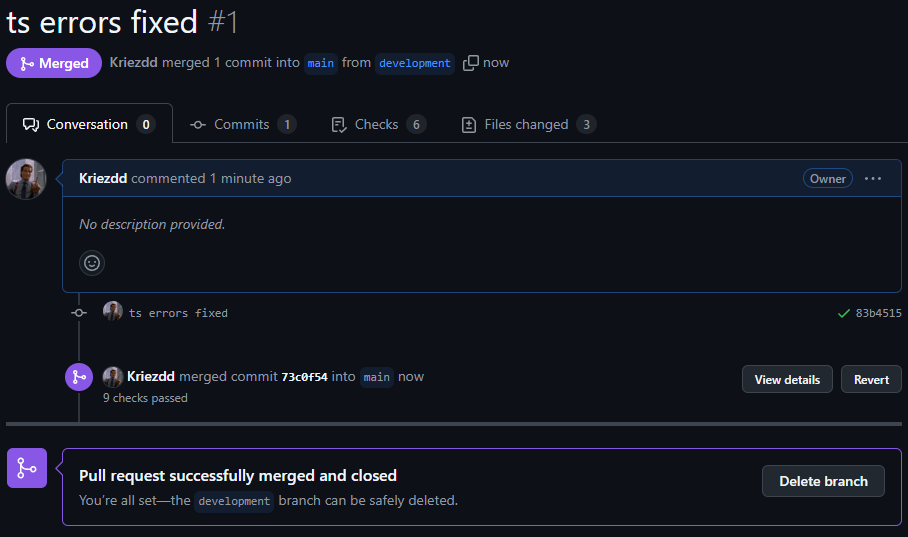
Отчёт о работе CI для коммита:



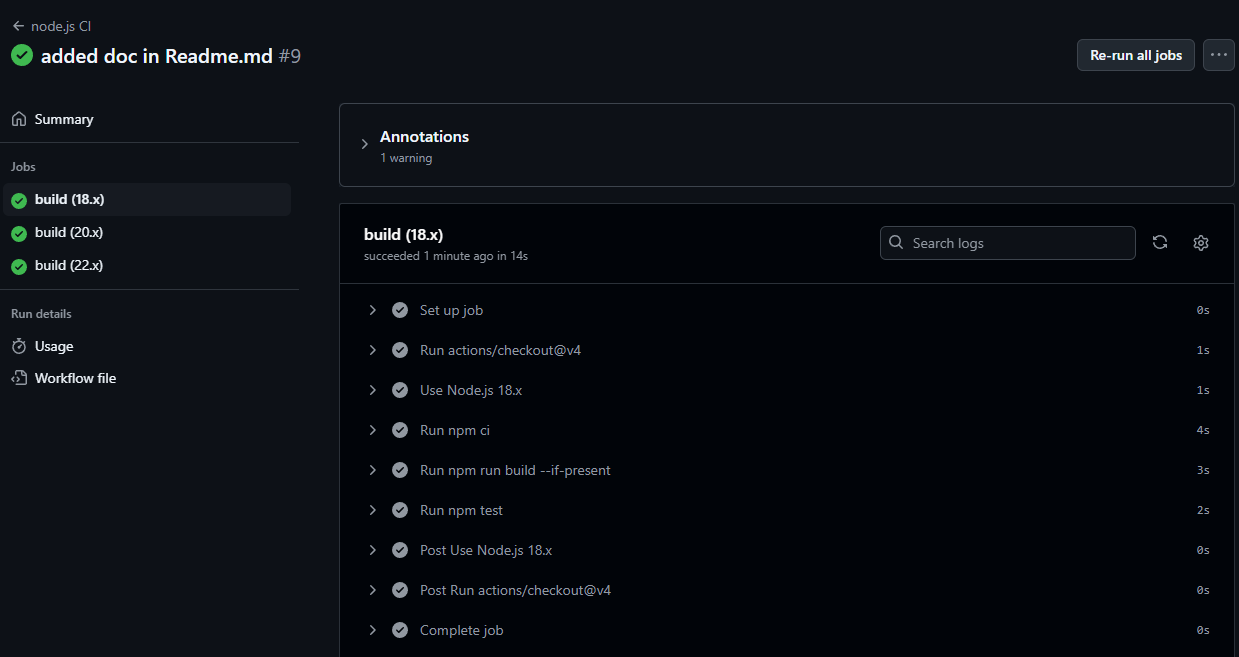
Тесты: 

1. Создайте Pull Request (PR) для вливания изменений из ветки development в основную ветку (например, main).
2. Настройте автоматическое тестирование и слияние PR при успешном прохождении всех тестов.
3. Проверьте, что изменения из ветки development успешно влились в основную ветку с помощью CI/CD.

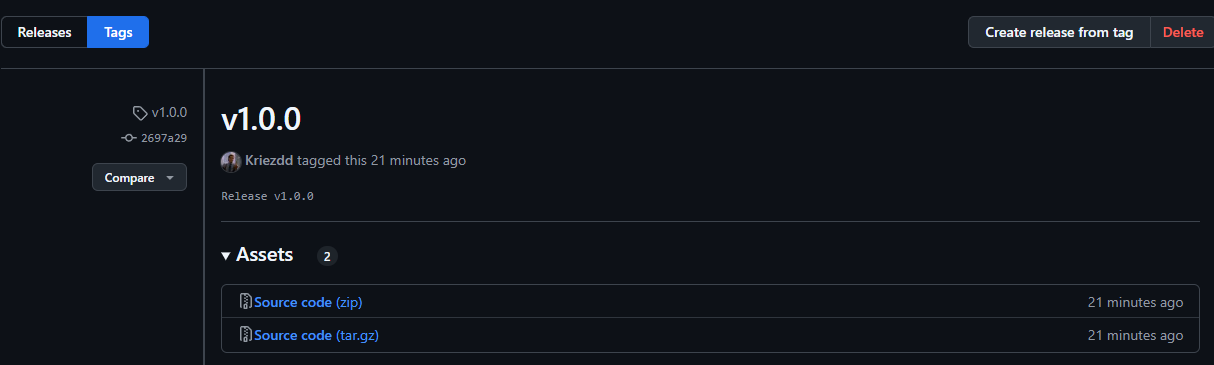




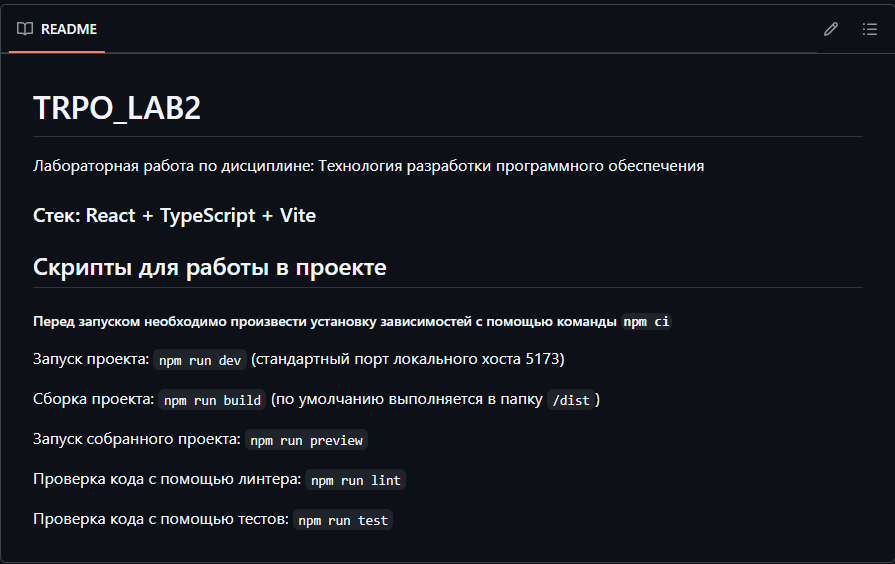
1. Создайте ветку release для подготовки к выпуску новой версии вашего проекта.
2. Подготовьте все необходимые изменения (например, обновление версии, исправление багов) в ветке release.
3. Убедитесь, что CI успешно запустился после коммита в ветку release и все тесты прошли успешно.



1. Создайте новый тег для версии вашего проекта и опубликуйте его на платформе Git.

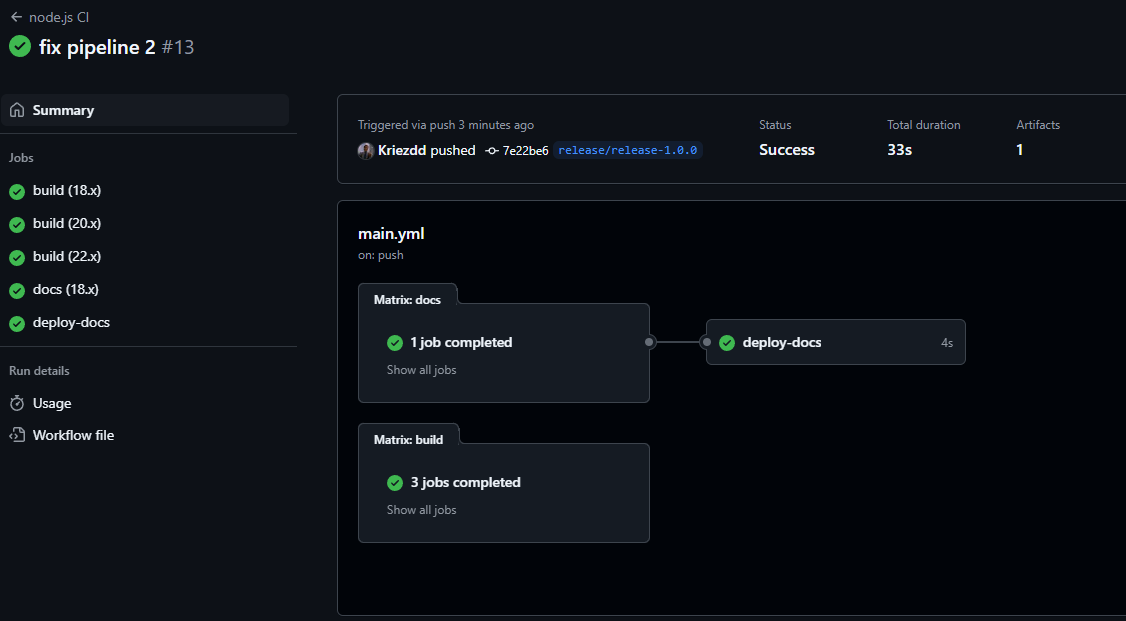


1. Создайте документацию к вашему проекту и добавьте ее в репозиторий.

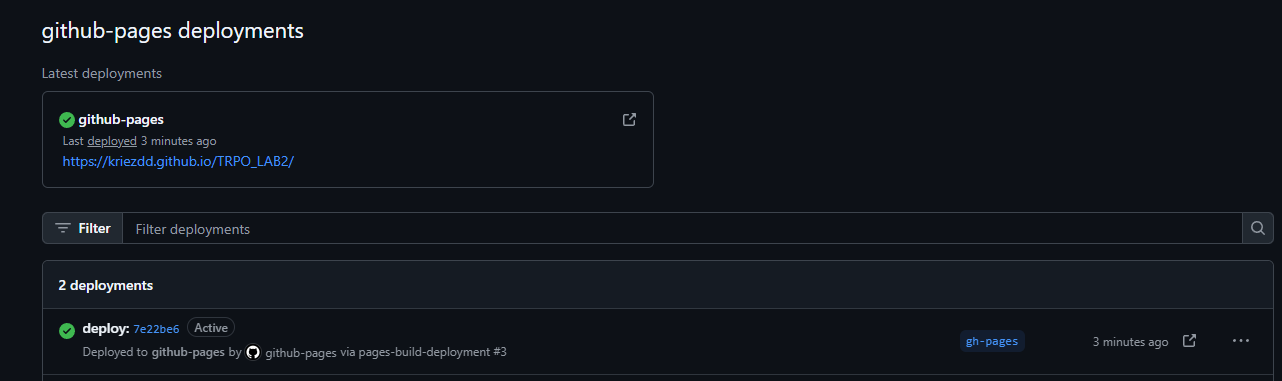


1. Настройте автоматическую генерацию документации при каждом обновлении кода через CI/CD.
2. Проверьте, что документация успешно обновляется при каждом изменении кода.
3. Заключите текущую версию проекта и завершите работу над лабораторной работой.

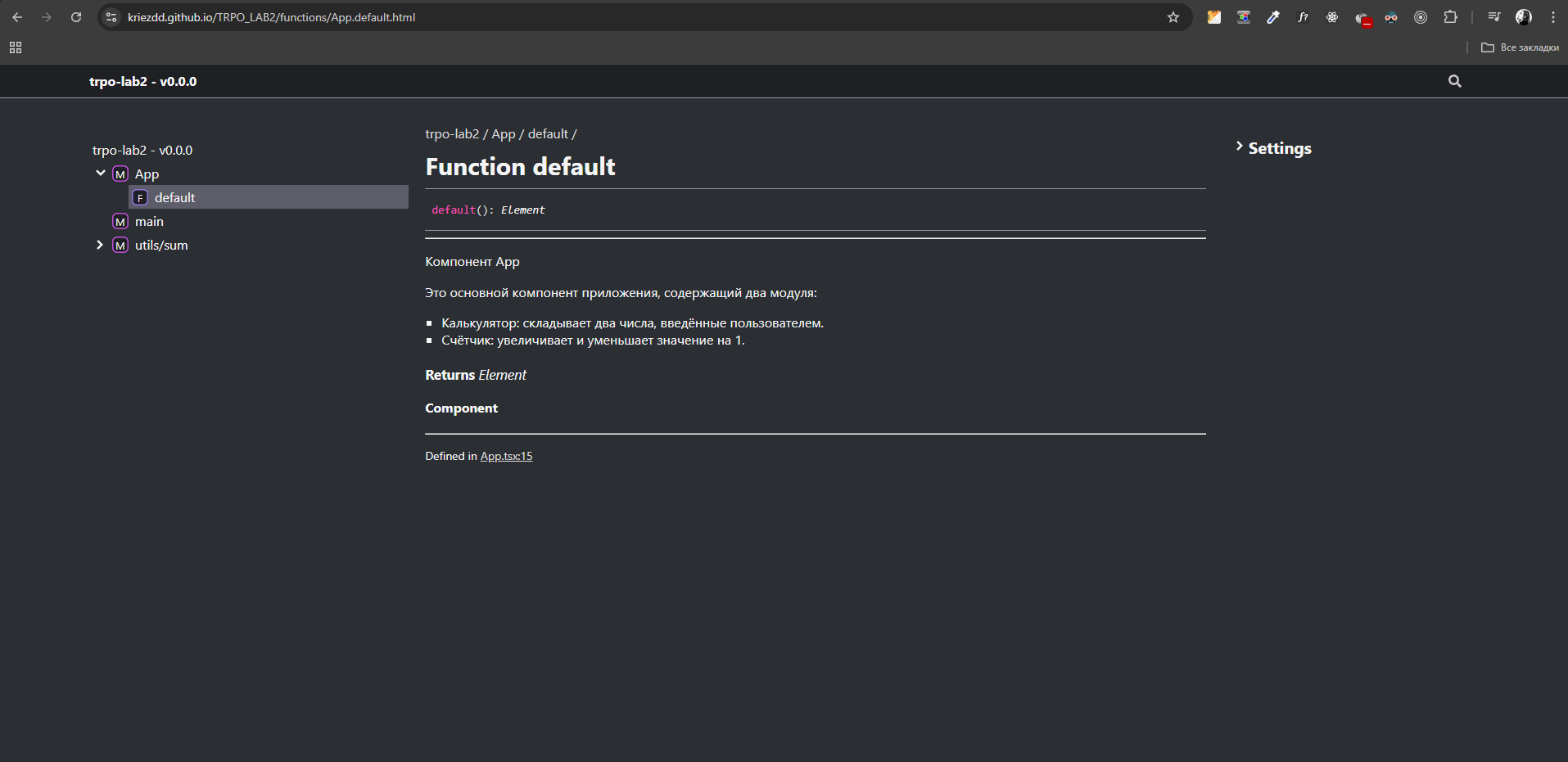




Авто-деплой документации на gh-pages:



Документация:



Ссылка на релиз ветку проекта:

https://github.com/Kriezdd/TRPO\_LAB2/tree/release/release-1.0.0