**ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ GIT+GITHUB ВО ВРЕМЯ ОБУЧЕНИЯ**

1. **Вариант с использованием организаций и команд**

Алгоритм работы по такому варианту имеет следующий вид:

1. Преподаватель создает организацию (имя может иметь подобный формат: <*имя группы*>.<*год*>) в Github с бесплатным тарифом. Подробнее см. на рисунках Рисунок 1 – Рисунок 4.

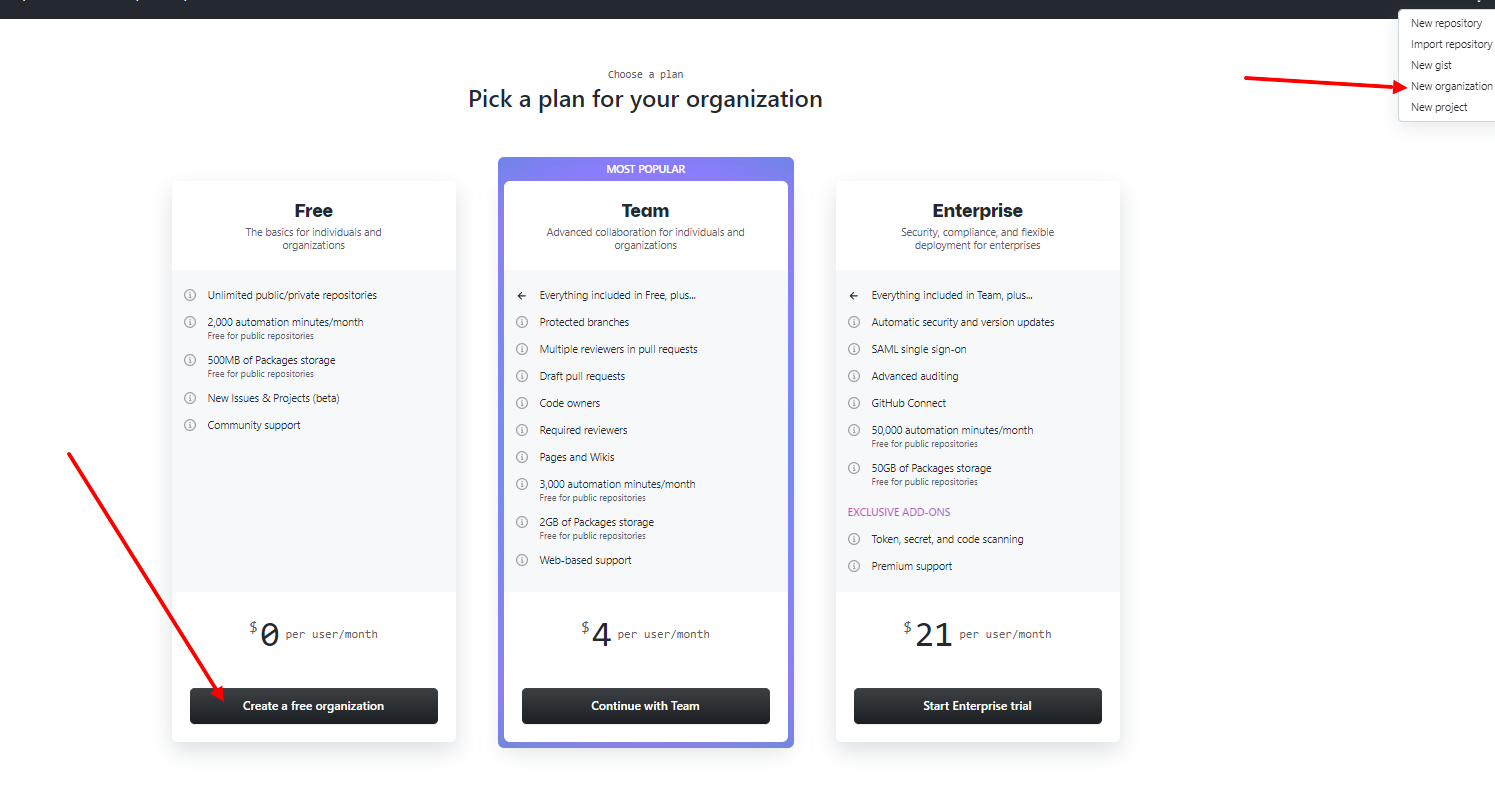


Рисунок 1 – Создание организации

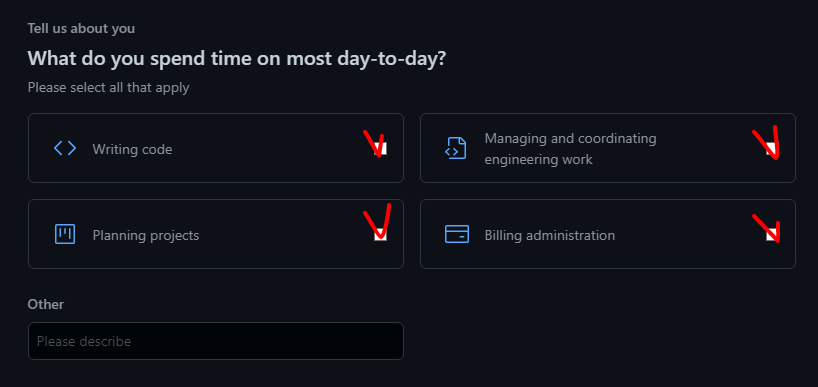


Рисунок 2 – Настройка организации. Часть 1.

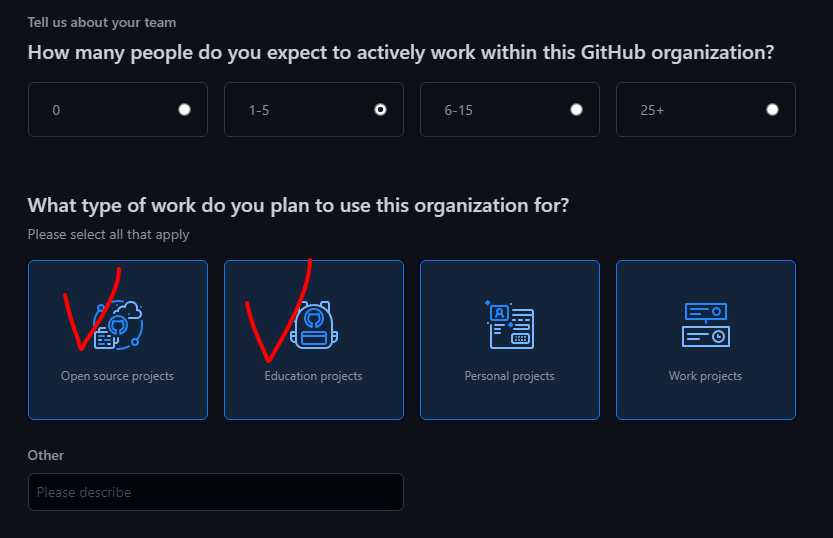


Рисунок 3 – Настройка организации. Часть 2.

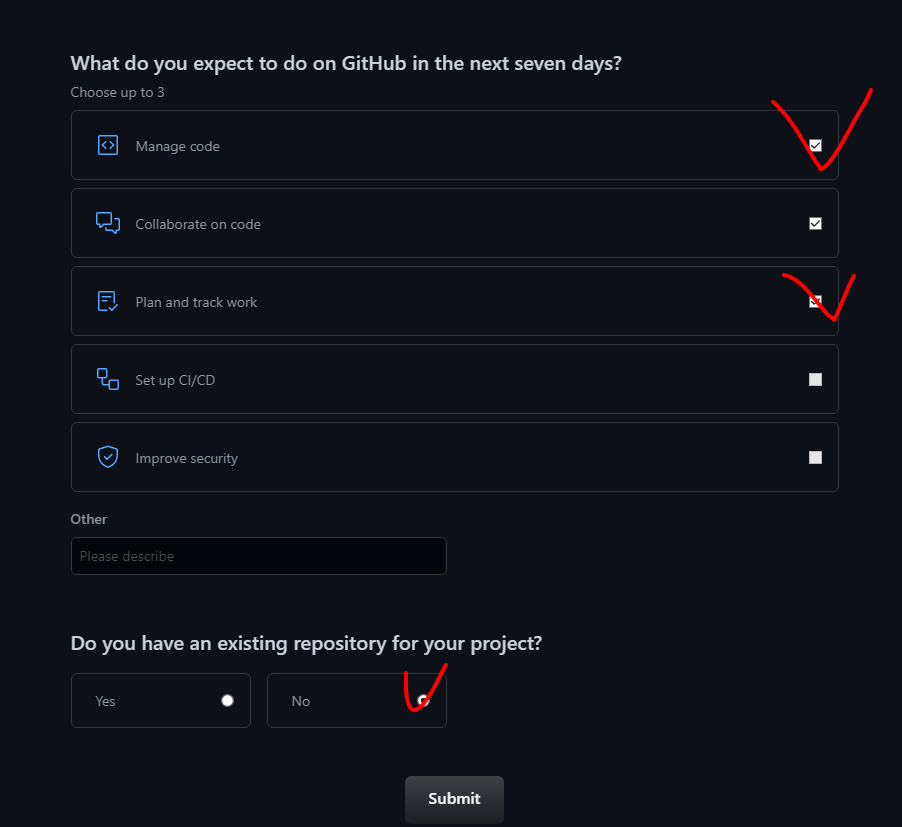


Рисунок 4 – Настройка организации. Часть 3.

1. Преподаватель переходит на страницу организации ([https://github.com/{*ИмяОрганизации*}](https://github.com/%7bИмяОрганизации%7d)) и приглашает пользователей (студентов) в организацию, как показано на рисунке Рисунок 5.

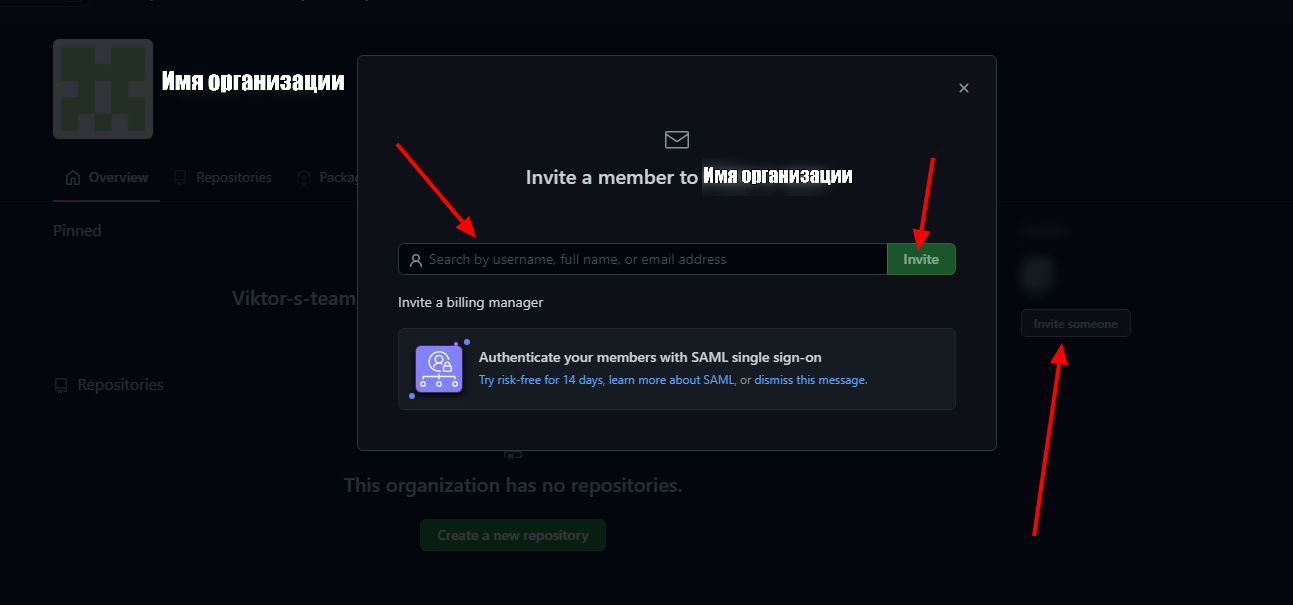


Рисунок 5 – Приглашение пользователей-студентов в организацию

1. Создается две (возможно более, по необходимости) команды. В одной команде – студенты, в другой преподаватель (преподаватели).
2. Создается учебный проект (проекты). В проекте должен быть один файл или одна директория для изменения на каждого студента.
3. Преподаватель разграничивает права для команд (<https://docs.github.com/en/organizations/managing-access-to-your-organizations-repositories/managing-team-access-to-an-organization-repository>), в частности, запрещается пуш в ветку мастер (мейн).
4. Преподаватель создает одну «рабочую» ветку для того, чтобы студенты могли пушить свои изменения.
5. Каждый студент выполняет задание только в своей директории.
6. После финального коммита, студент делает запрос на слияние через pull request из «рабочей» ветки в ветку мастер (мейн). Всего таких пулл-реквестов будет N, где N – число студентов, выполняющих задание.
7. По итогу преподаватель может оценивать работу каждого студента путем принятия/отклонения пулл реквеста, а также преподаватель может писать свои комментарии в issue.

**Плюсы данного варианта:**

* Студенты сразу же учатся работать в команде в условиях, приближенных к реальной коммерческой разработке.
* Преподавателю удобно раздавать задачи и назначать их студентам, а также устанавливать сроки для их выполнения.
* Данный метод хорошо сочетается с типовыми (для всех студентов) заданиями. Например, когда каждый студент разрабатывает свою HTML-страничку или же свой какой-то набор функций или методов класса.
* Автоматизация для преподавателя – многие организационные действия однократны (создание организации, команд, добавление пользователей)

**Минусы данного варианта:**

* Студенту может показаться сложным данный подход.
* Могут возникнуть сложности при работе со специфическими проектами.
* Данный вариант непригоден для уникальных заданий.

1. **Индивидуальные репозитории**

Данный вариант основан на том, что каждый студент сам создаст репозиторий под конкретное задание и добавляет в соавторы преподавателя, который осуществляет контроль разработки.

**Плюсы данного варианта:**

* Данный вариант хорошо подходит для уникальных проектов – курсовые и дипломные работы.
* Требуется меньше действий для начала работы.
* Студент учится самостоятельно работать с ветками и т.п.

**Минусы данного варианта:**

* Преподавателю приходится добавляться в каждый репозиторий.
* Потенциально большая вероятность отсутствия единого стиля оформления репозитория (затрудняет работу с кучей репозиториев).

1. **Работа с форками**

Вариант аналогичен первому, за исключением создания организации с командами. Вместо этого каждый студент делает форк базового проекта, а затем создает запрос на изменение в базовый проект, в ветку мастер.

Данный вариант имеет все те же достоинства и недостатки, что и первый вариант, за исключением сложности и работы в условиях, приближенных к коммерческой разработке.

**Примечание**: не рассматривались автоматизированные варианты проверки заданий через github, как это реализовано в RS School: https://app.rs.school/