БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ ХАНТЫ-МАНСИЙСКОГО АВТОНОМНОГО ОКРУГА - ЮГРЫ  
«Сургутский государственный университет»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Математическое и информационное обеспечение систем управления деятельностью предприятий нефтегазовой отрасли

ПРАКТИЧЕСКАЯ РАБОТА № 1

по теме: «GitHub - формируем проект»

по дисциплине: «Управление ИТ-проектами»

|  |  |  |
| --- | --- | --- |
|  | | Руководитель работы,  \_\_\_\_\_\_\_\_\_\_\_\_/Чалей И.В.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |
|  | | Автор работы,  Студент группы 601-41м  \_\_\_\_\_\_\_\_\_\_\_\_/Вильданова Л.А.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |
|  |  |

Сургут - 2025

ПРАКТИЧЕСКАЯ РАБОТА № 1

по теме: «GitHub - формируем проект»

Методические рекомендации по созданию проекта на GitHub

В современном мире управление ИТ-проектами невозможно представить без использования инструментов коллективной разработки и контроля версий. Одним из наиболее популярных и удобных сервисов является **GitHub**. Он сочетает в себе систему управления версиями Git и веб-платформу для совместной работы над проектами.

GitHub используется не только для хранения исходного кода, но и для организации командной работы: постановки задач, контроля их выполнения, обсуждения идей, документирования и автоматизации процессов разработки. С его помощью студенты и профессиональные разработчики могут создавать проекты любой сложности — от учебных приложений до масштабных программных систем.

Применение GitHub в учебном процессе позволяет сформировать у студентов практические навыки работы с современными инструментами управления проектами, научить их распределять задачи, контролировать прогресс, а также документировать цели и результаты работы. Освоение этого инструмента становится важным шагом в подготовке будущих специалистов к реальной работе в ИТ-командах.

Для начала работы на GitHub необходимо создать **репозиторий**.

**Репозиторий** – это хранилище, в котором содержатся все материалы проекта: исходный код, документация, файлы конфигурации, данные, а также история изменений. Именно репозиторий является центральным объектом работы на GitHub, поскольку он обеспечивает хранение и версионность файлов проекта, возможность совместной работы нескольких участников, фиксацию изменений в виде коммитов, ведение документации и структуры проекта, постановку и отслеживание задач через систему Issues, создание веток (branches) для разработки отдельных функций без риска повредить основную версию проекта.

Репозиторий можно представить как «папку проекта», дополненную инструментами для контроля версий и совместной работы. Он может быть:

**- публичным (Public)** – доступен всем пользователям GitHub;

**- частным (Private)** – доступ имеют только автор и приглашённые участники.

## **Создание нового репозитория**

## *Шаг 1. Переход к созданию репозитория*

- Войдите в аккаунт GitHub.

- Откройте вкладку **Repositories**.

- В правом верхнем углу нажмите кнопку **New**.

- Откроется окно **Create a new repository**.

## *Шаг 2. Заполнение формы «Create a new repository»*

### Общие сведения (General):

**- Owner**\* — выберите владельца (личный профиль или организацию).

**- Repository name**\* — введите название проекта. Требования: латиница, цифры, дефисы/подчёркивания/точки, без пробелов; имя должно быть **уникально** в рамках выбранного владельца; пишите кратко и понятно.

**- Description** — кратко опишите назначение проекта (необязательное поле).

### Конфигурация (Configuration):

**- Choose visibility**\* — укажите тип доступа:

* **Public** — репозиторий виден всем пользователям.
* **Private** — доступ только у вас и приглашённых участников.

**- Add README** — включите опцию для автоматического создания файла **README.md** (настоятельно рекомендуется). Этот файл будет «визитной карточкой» проекта: цели, задачи, запуск, структура и т.д.

**- Add .gitignore** — выберите шаблон .gitignore. - это текстовый файл со списком масок файлов и каталогов, которые Git не отслеживает (не добавляет в коммиты). Выберите шаблон под ваш стек (Python, Node, Java, Xcode и др.) и при необходимости дополните позже.

**- Add license** — выберите лицензию (рекомендуется). Назначение: определить, **как другие могут использовать ваш код** (копировать, модифицировать, распространять). Если оставить **No license**, формально действует режим «все права защищены»: код смотреть можно (если репозиторий публичный), но **использовать/копировать/модифицировать нельзя** без вашего разрешения. Учебные/открытые проекты — чаще всего используют **MIT** или **Apache-2.0.**

Нажмите **Create repository, и будет** создан репозиторий с выбранной видимостью, при включённых опциях — будут добавлены файлы README.md, .gitignore, LICENSE.

1. **Заполнение файла README.md**

Непосредственно после создания репозитория следует подготовить и заполнить файл README.md. На практике данный шаг нередко игнорируется, однако README.md выступает «визитной карточкой проекта»: формирует первичное впечатление у пользователей и контрибьюторов, снижает количество повторяющихся запросов по эксплуатации и участию в развитии проекта (в том числе со стороны будущего «самого автора»), а также обеспечивает распознаваемость, корректную интерпретацию и последующее использование проекта профессиональным сообществом.

## **Минимально необходимое содержание README.md**

1. **Название проекта** — краткое и однозначно интерпретируемое.
2. **Краткое описание (назначение)** — 1–3 предложения о целевой аудитории и полезности.
3. **Цели и задачи** — планируемые результаты и функциональные блоки.
4. **Технологии и требования** — языки, фреймворки, версии и ключевые зависимости.
5. **Установка и запуск** — последовательность команд для развертывания и проверки работоспособности.
6. **Краткая структура проекта** — основные каталоги и их функциональная роль.
7. **Лицензия** — ссылка на файл LICENSE или явное указание типа лицензии.

Игнорирование подготовки README.md сопоставимо с возведением здания без адресной таблички: даже при высоком качестве исходного кода его практическая ценность, воспроизводимость и потенциал повторного использования остаются ограниченными из-за отсутствия ясного контекста и инструкций.

### **Формирование структуры проекта**

Структура проекта — это иерархическая организация каталогов и файлов, отражающая логическую архитектуру программного обеспечения, данных и сопровождающей документации. В среде GitHub, основанной на системе контроля версий Git, формирование структуры выступает не технической формальностью, а **фундаментальным этапом проектирования**. Инициализация базовой структуры осуществляется при создании репозитория и далее **эволюционирует итеративно** вместе с развитием проекта. В отличие от локальной файловой системы, структура в GitHub **неразрывно связана с историей изменений** (commit history), что придаёт ей дополнительное — временное — измерение.

**Цели формирования продуманной структуры.**

1. **Повышение находимости и понимания.** Логичная группировка артефактов (например, исходный код — в ./src, документация — в ./docs, тесты — в ./tests) снижает когнитивную нагрузку на новых участников и ускоряет ориентацию в кодовой базе, что напрямую соотносится с принципами поддерживаемости.
2. **Обеспечение масштабируемости.** Модульная организация, заложенная на старте, позволяет добавлять новые функциональные блоки без радикальной реорганизации существующих компонентов — критически важное требование для долгосрочных академических и коммерческих проектов.
3. **Поддержка автоматизации и интеграции.** Чёткое разделение по назначению является предпосылкой корректной настройки средств непрерывной интеграции и поставки (CI/CD). Скрипты и пайплайны могут быть нацелены на конкретные каталоги (например, выполнение npm install в подкаталоге ./frontend), что повышает воспроизводимость и уменьшает избыточные операции.

**Процесс формирования:**  
 Структура формируется **итеративно**: базовые каталоги задаются на этапе инициализации репозитория и уточняются по мере появления новых требований. Практическая реализация возможна двумя взаимодополняющими способами:

* через интерфейсы командной строки Git (в том числе интегрированные в среду разработки);
* непосредственно через веб-интерфейс GitHub (Code → Add file → Create new file), где каталоги создаются путём добавления в них первого файла (например, docs/README.md или src/.gitkeep), поскольку пустые папки системой не хранятся.

Такое понимание и строгое соблюдение принципов структурирования обеспечивает прозрачность, поддерживаемость и масштабируемость проекта на всех стадиях его жизненного цикла.

1. **Добавление участников в проект**

Совместная работа над проектом на GitHub требует, чтобы руководитель мог приглашать новых членов команды и распределять между ними задачи.

**1. Добавление участников в репозиторий**

1. Руководитель проекта открывает страницу репозитория на GitHub.
2. Переходит в раздел Settings → Collaborators.
3. Нажимает кнопку Add people (значок «плюс»).
4. Вводит **GitHub-никнейм** или **адрес электронной почты** человека. Достаточно знать почту или ник — GitHub сам найдёт аккаунт.
5. Подтверждает выбор кнопкой Add <ник>.
6. GitHub отправляет приглашение участнику.

## **Принятие приглашения в проект**

После того как руководитель проекта отправил приглашение, участнику необходимо его принять. Приглашение приходит сразу в два места:

**- На GitHub (в веб-интерфейсе):**

1. Приглашение отображается прямо на странице репозитория, куда вас пригласили.
2. Пока оно не принято, в верхней части страницы будет виден баннер с предложением присоединиться к проекту.

**- На электронную почту:**

1. GitHub отправляет письмо на e-mail, привязанный к аккаунту приглашённого пользователя. В письме указывается, кто именно пригласил, и к какому репозиторию предоставляется доступ.
2. Внизу письма есть синяя кнопка «**View invitation»**. При нажатии на неё пользователь переходит на страницу GitHub, где уже доступна кнопка **Accept invitation** (принять) или **Decline** (отклонить).
3. Приглашение действительно ограниченное время (обычно 7 дней).

После того как приглашение принято, участник получает права **Collaborator** — возможность читать и писать код в репозитории, а также быть назначенным на задачи (**Issues**).

1. **Список задач (Issues)**

Issues — это единицы планирования и контроля работ. Они фиксируют требования и изменения, обеспечивают декомпозицию целей проекта, прослеживаемость решений и коммуникацию внутри команды. Создание новой задачи: перейдите в вкладку Issues → New issue.

## **Что должно быть указано в каждой задаче (Issue):**

* **Заголовок** — кратко по сути действия и результата.
* **Контекст и цель** — зачем выполняется работа; связь с целями проекта.
* **Ожидаемый результат**  — что должно появиться/измениться.
* **Критерии приёмки**  — проверяемые пункты чек-листом.
* **Шаги/подзадачи** — основные этапы выполнения.
* **Зависимости и риски** — на что опирается задача, что может блокировать.
* **Оценка/сложность** (при применении методики) — S/M/L.
* **Метки** - для классификации и фильтрации задач руководитель назначает метки, например good first issue, enhancement и т.д.

## **Назначение исполнителей и выбор задач участниками:**

### - Если участник **включён** в проект (collaborator), тогда участник вправе **самоназначиться** через поле **Assignees** (если это разрешено политикой проекта) или руководитель назначает исполнителя сам. Issue → **Assignees → себя**.

### - Если участник ****не включён**** в проект (не является collaborator), тогда самоназначение недоступно: поле **Assignees** неактивно. Тогда нужно оставить **комментарий** под задачей с запросом: “Готов(а) выполнить задачу или Прошу назначить”.

# **Создание форка (Fork)**

**Fork** (форк) — это полная копия существующего репозитория, создаваемая в вашем личном аккаунте или организации на GitHub. Форк используется для независимой работы над проектом без внесения изменений в оригинальный (исходный) репозиторий. Это базовый механизм участия в сторонних проектах, выполнения заданий без доступа к оригиналу, а также экспериментов с чужим кодом.

Таким образом, вы получаете **полноценную копию всего проекта**, а не только своих задач или фрагментов кода.

## **Назначение форка:**

1. Выполнение заданий на основе чужого проекта, без риска навредить оригиналу.
2. Участие в чужом проекте без прав записи.
3. Подготовка предложений (через Pull Request) к оригинальному проекту.
4. Создание учебных или экспериментальных версий проекта.

## **Как создать форк репозитория:**

В правом верхнем углу страницы нажмите кнопку **Fork** (обозначена иконкой в виде вилки). В открывшемся окне выберите, в какой аккаунт будет скопирован репозиторий — это может быть ваш личный профиль или организация, участником которой вы являетесь. При необходимости вы можете задать новое имя для форка и добавить краткое описание. После этого нажмите кнопку **Create fork**.

По завершении процесса вы автоматически будете перенаправлены в созданный репозиторий, расположенный по адресу: https://github.com/<ваш\_профиль>/<название\_форка>