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 № 3

по теме: «Проект управление процессом подготовки магистранта»

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 № 3

по теме: «Проект управление процессом подготовки магистранта»

Для того чтобы **эффективно и наглядно отслеживать выполнение домашних заданий**, а также с возможностью **контролировать все предыдущие версии работ магистрантов**, предлагаю использовать систему **GitHub**.

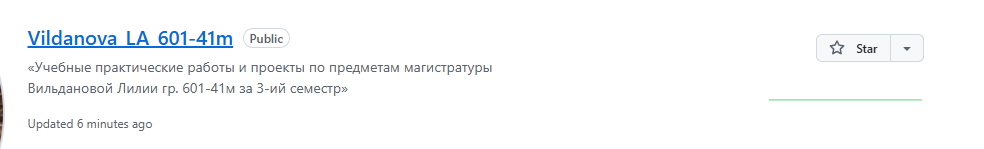
Для каждого магистранта следует **создать отдельный репозиторий**, куда он будет выкладывать свои домашние задания. Такой подход позволит:

1. удобно отслеживать прогресс каждого студента;
2. сохранять все версии выполненных заданий;
3. обеспечивать прозрачность и системность в процессе проверки.

**Пошаговые действия:**

1. На первом этапе необходимо **создать новый репозиторий** на платформе **GitHub**.  
   Рекомендуется назвать его по **фамилии студента**, добавив для удобства идентификации **номер группы** (например, Vildanova\_LA\_601-41m).

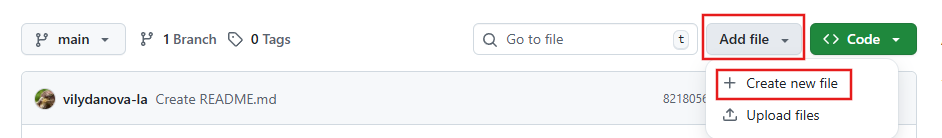
После создания **устанавливаем видимость репозитория как **публичную**** — это упростит процесс проверки и совместной работы. Д**обавляем в участники (collaborators)** соответствующего студента, чтобы он имел возможность самостоятельно загружать свои домашние задания и вносить изменения.



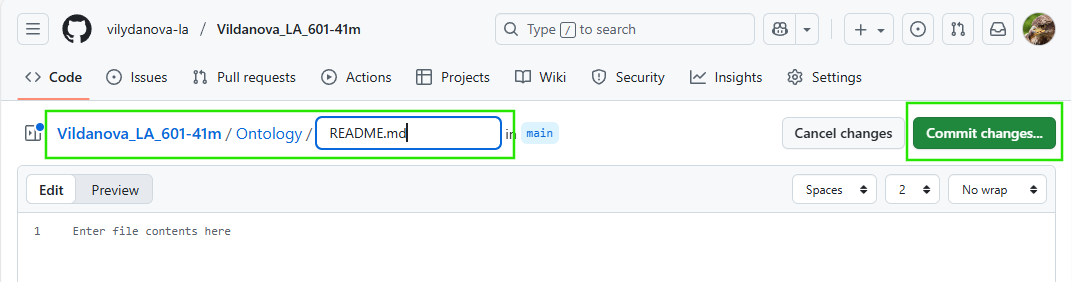
1. На следующем этапе необходимо **создать три папки**, соответствующие учебным дисциплинам:

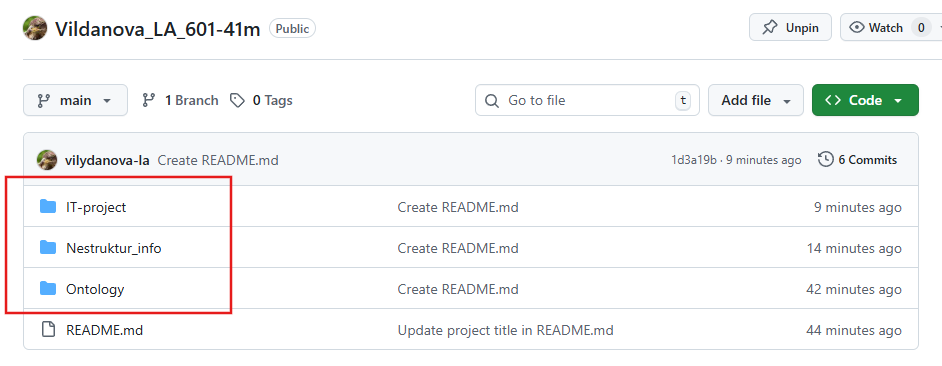
* Онтология
* Управление ИТ-проектами
* Управление неструктурированной информацией

Для этого в репозитории можно воспользоваться функцией **создания файла** через интерфейс GitHub (Add file → Create new file).

  
 При создании каждой папки следует указать путь в формате: Название\_папки/README.md

Это важно, так как GitHub **не создаёт пустые папки** — наличие файла README.md позволит сохранить структуру каталогов.  
В дальнейшем в эти папки студент будет загружать файлы с домашними заданиями по соответствующим предметам.

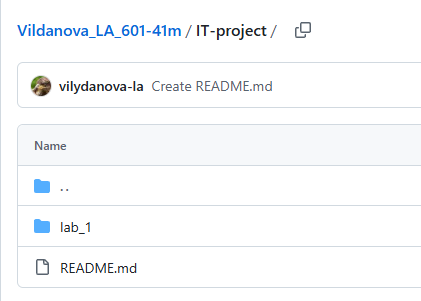




1. После создания основных папок для дисциплин необходимо **организовать структуру внутри каждой из них**.  
    Рассмотрим пример для предмета «IT-project»:

* Заходим в папку предмета IT-project в репозитории.
* Создаём новую папку с названием, соответствующим лабораторной работе, например: Lab\_1/README.md (обязательно добавляем файл README.md, поскольку GitHub **не позволяет создавать пустые папки)**.

Этот файл может содержать краткое описание лабораторной работы — её цель, задание и статус выполнения.



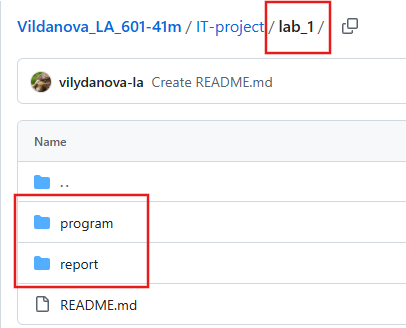
1. Далее, внутри папки Lab\_1 необходимо **создать дополнительные вложенные каталоги**, чтобы структурировать материалы по типам файлов.

Рекомендую создать следующие папки:

* **program/** — сюда будет помещаться исходный **код программы**, написанной студентом;
* **report/** — в эту папку студент будет **загружать отчёт по лабораторной работе** (в формате .pdf, .docx или другом установленном преподавателем).

Создавать их следует аналогичным образом — через добавление файла README.md внутри каждой папки.

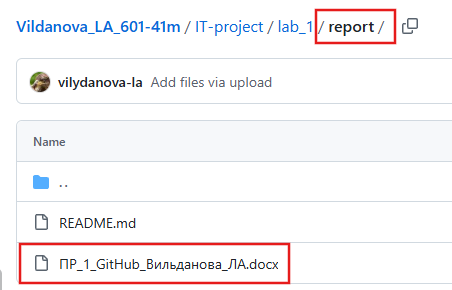
Такая организация сделает репозиторий **аккуратным, удобным для проверки и расширяемым** при добавлении новых лабораторных работ.



1. Далее **добавим отчёт по первой лабораторной работе**. Для этого выполняем следующие шаги:

* Переходим в репозитории по пути: IT-project/Lab\_1/report
* Нажимаем кнопку **Add files → Upload files**.
* Перетаскиваем на поле файл с отчётом по работе (в формате **.docx** или **.pdf**) либо выбираем его вручную с компьютера.
* После загрузки нажимаем **Commit changes** — чтобы изменения сохранились в репозитории.

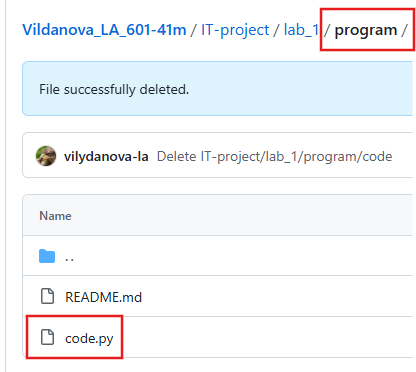
Отчёт будет добавлен в соответствующую папку, и преподаватель сможет легко его просмотреть или скачать при проверке.



1. После загрузки отчёта необходимо **добавить исходный код программы**, выполненной в рамках лабораторной работы. Для этого выполняем следующие шаги:

* Переходим в репозитории по пути: IT-project/Lab\_1/program
* Нажимаем **Add files → Create new file**.
* В открывшемся окне вставляем **текст программы** (код), написанный студентом.
* В поле имени файла указываем, например: code.py
* После вставки кода нажимаем **Commit changes** для сохранения файла в репозитории.

Таким образом, исходный код будет аккуратно размещён в папке program, а преподаватель сможет легко его просмотреть и при необходимости прокомментировать.



1. После того как структура для одного предмета (например, IT-project) успешно создана и заполнена, аналогичные действия необходимо **выполнить для всех остальных дисциплин** — Онтология и Управление неструктурированной информацией.

Для каждой из этих папок формируется такая же структура:

Название\_предмета/

├── Lab\_1/

│ ├── program/

│ │ └── README.md

│ ├── report/

│ │ └── README.md

│ └── README.md

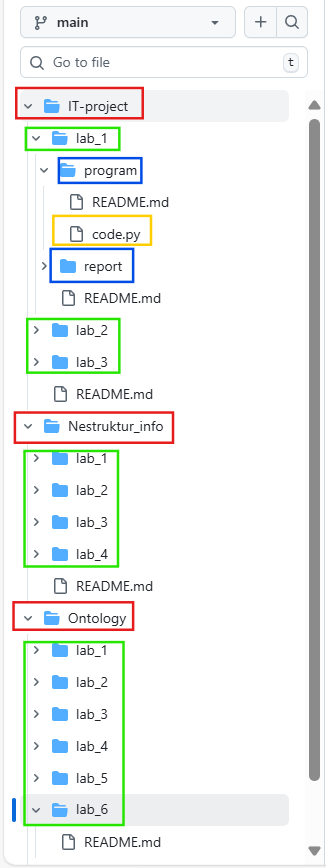
├── Lab\_2/

│ ├── program/

│ ├── report/

│ └── README.md

└── README.md



В дальнейшем **преподаватель** размещает в соответствующих папках **задания** (например, в виде текстовых файлов или описаний в README.md), где указывает цель, требования и критерии оценки работы.  
 **Студент**, в свою очередь, добавляет в эти папки:

* в report/ — отчёт по выполненному заданию;
* в program/ — исходный код программы или другие необходимые файлы.

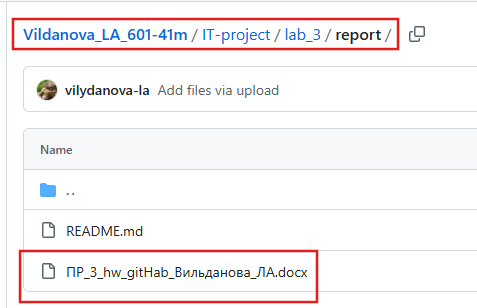
Такая система работы через GitHub обеспечивает:

* прозрачность и удобство взаимодействия между преподавателем и студентом;
* сохранность всех версий заданий и ответов;
* возможность легко отслеживать прогресс выполнения работ каждым магистрантом;
* единообразную и понятную структуру хранения данных по всем дисциплинам.

В результате преподаватель получает полный контроль над учебным процессом, а студент — удобный инструмент для систематизации и представления своих результатов.

1. Для примера, отчёт по третьей лабораторной работе по дисциплине «IT-project» находится по следующему пути в GitHab: vilydanova-la/Vildanova\_LA\_601-41m/IT-project/lab\_3/report

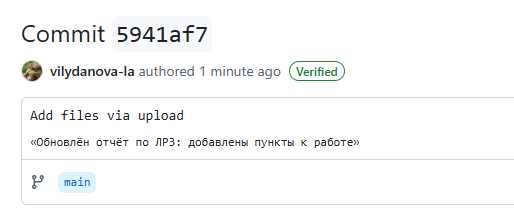
По указанному адресу можно найти загруженный файл отчёта, а также при необходимости ознакомиться с кодом программы, расположенным в соседней папке program.



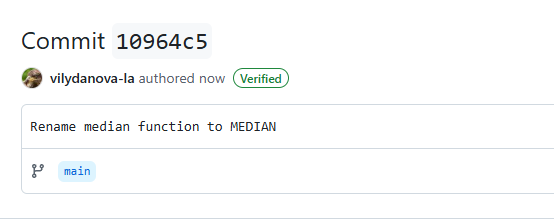
1. Если преподаватель отправляет работу на доработку, он может **оставить комментарии и рекомендации прямо в файле** README.md внутри соответствующей папки (например, Lab\_3/report/).  
    Студент, получив замечания, должен внести исправления и **заменить старую версию отчёта на обновлённую**, при этом сохранив историю изменений в репозитории:

* Перейдите в папку, где находится файл, например: IT-project/Lab\_3/report/
* Нажмите **Add file → Upload files**.
* Перетащите в окно новую версию отчёта с тем же именем файла, например: ПР\_3\_hw\_GitHub\_Вильданова\_ЛА.docx
* GitHub автоматически определит, что файл с таким именем уже существует, и предложит **заменить его (overwrite)**.
* В нижней части страницы обязательно добавьте **комментарий к коммиту**, например: «Обновлён отчёт по ЛР3: добавлены пункты к работе»
* Нажмите зелёную кнопку **Commit changes**, чтобы сохранить обновления.

**Примечание:**  
 Комментарии к коммитам обязательно пишутся для файлов в форматах **Word (.docx) или PDF,** так как GitHub не может отображать текстовые изменения внутри этих типов файлов.  
 Таким образом, комментарии служат кратким описанием внесённых правок и помогают преподавателю быстро понять, что именно было изменено.



1. Если был изменён ****код программы**,** то эти изменения **видно в коммите** — GitHub автоматически **отслеживает и отображает различия** между старой и новой версиями файла.  
    Система построчно показывает, что было **добавлено, удалено или изменено**, что делает процесс проверки кода максимально прозрачным.



GitHub самостоятельно отслеживает изменения для **текстовых форматов**, таких как:

* .py — Python
* .java — Java
* .cpp, .c, .h — C и C++
* .js, .ts — JavaScript / TypeScript
* .html, .css — веб-разметка и стили
* .md — Markdown (включая README-файлы)
* .txt и другие **текстовые файлы**.

А вот для **бинарных форматов** — например, .docx, .pdf, .xlsx, .jpg, .png — GitHub **не отображает различия внутри файла**, поэтому при обновлении таких файлов важно **добавлять комментарий к коммиту**, описывающий внесённые изменения.