**Отчет об выполнении задания №3**

Выполнена регистрация в сервисах GitHub и GitLub.

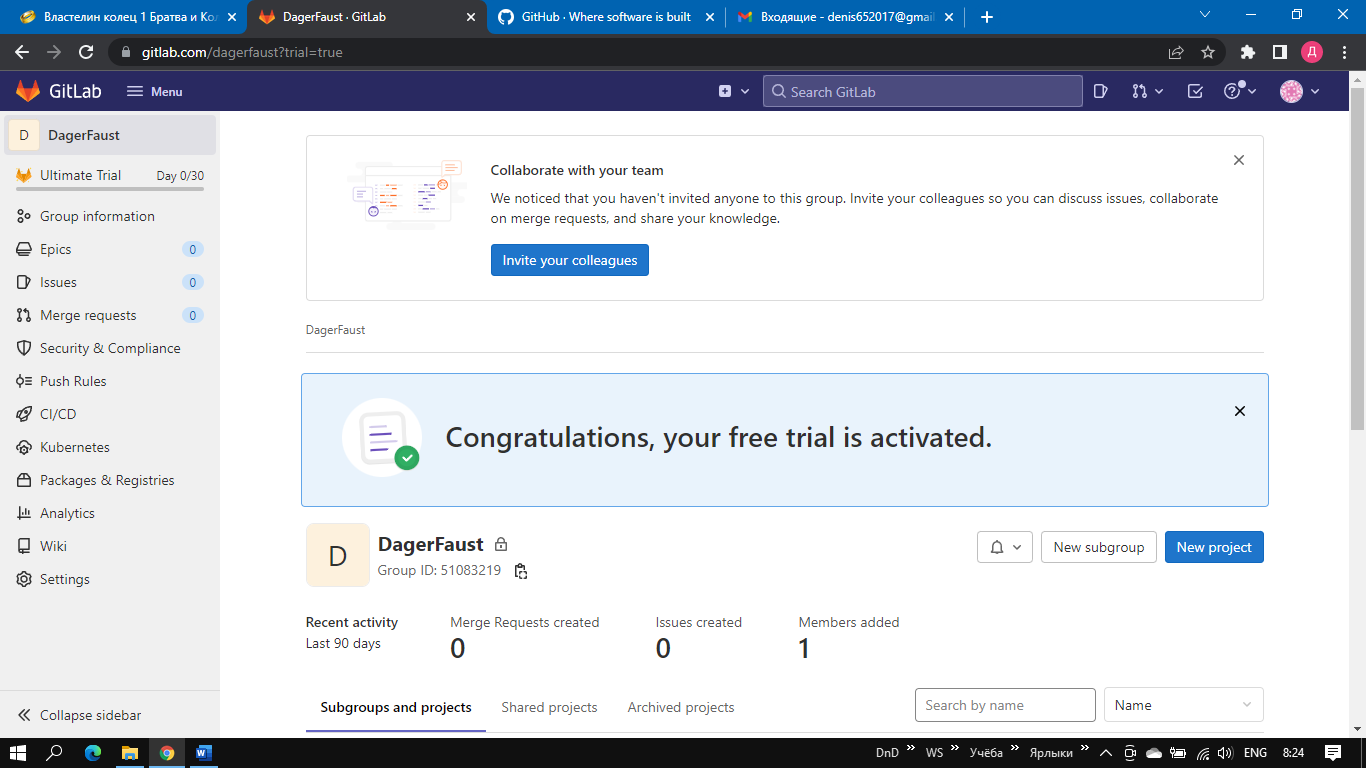


Рисунок №1 – Завершение регистрации на GitLub

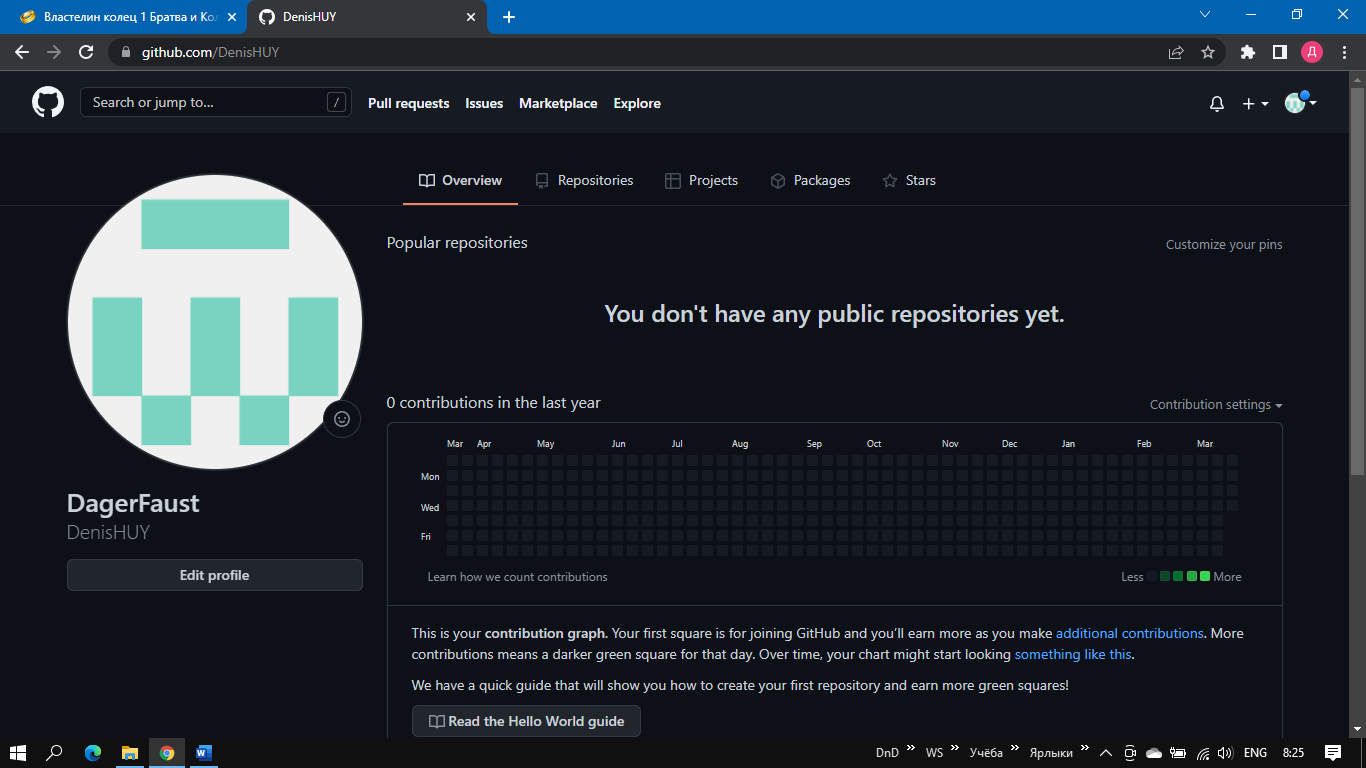


Рисунок №2 – Завершение регистрации на GitHub

Система контроля версий – это обязательный инструмент в арсенале программиста любого уровня: от новичка, который только осваивается в профессии, до тимлида, чей опыт исчисляется многими годами и проектами.

**Принципы работы СКВ**

многие представляют себе, каким образом вносятся изменения в тот или иной проект в процессе работы. То есть, тут стоит задача не потерять существующую работоспособную версию проекта. Для этого обычно создается новая папка (чаще всего ее так и называют «Новая папка», дополняя имя, к примеру, датой или парой символов), в которую вы копируете имеющуюся версию и все дальнейшие изменения производите уже именно с ней.

Подобных папок может набраться довольно много. Из-за этого в какой-то момент усложняется откат к предыдущим версиям, труднее становится контролировать и упорядочивать вносимые изменения и т.д. А если проект ведут несколько специалистов, картина усугубляется.

Системы контроля версий (СКВ или VCS) разработаны специально для того, чтобы максимально упростить и упорядочить работу над проектом (вне зависимости от того, сколько человек в этом участвуют). СКВ дает возможность видеть, кто, когда и какие изменения вносил; позволяет формировать новые ветви проекта, объединять уже имеющиеся; настраивать контроль доступа к проекту; осуществлять откат до предыдущих версий.

**Задачи СКВ**

В целом задачи для системы контроля версий уже описаны выше, но можно пояснить еще, для тех, кто сталкивается с термином впервые. Для наглядности представьте, что работа над проектом – это игра, которую нужно пройти, двигаясь от одной контрольной точки – к другой (это и будут промежуточные версии проекта).

Например, в процессе работы заказчик замечает баг, который непременно должен быть устранен. Для этого разработчику достаточно вернуться к нужному коммиту и исправить в нем ошибку. После этого остается лишь обновить основную ветку.

Но чего можно ожидать, если программист на своем компьютере не использовал систему контроля версий и не сохранял создаваемые промежуточные копии плагина? Специалист будет переделывать проект практически заново. Возможно, ему и удастся внести нужные изменения, не нарушив основную логику, но это будет очень и очень сложно.

**GIT**

Современный программист вне зависимости от его профиля должен знать, как работать с GIT. Да, у разных СКВ есть свои достоинства и недостатки, но большая часть компаний на сегодняшний день используют именно систему контроля версий GIT.

GIT относится к распределенным системам контроля версий, и как децентрализованная СКВ имеет свои преимущества:

* сетевых задержек не бывает, поэтому все операции выполняются очень быстро;
* идеально подходит для проектов, к работе над которыми привлекается целая команда специалистов;
* возможность сохранить данные в случае выхода из строя центрального сервера.

Версии хранятся тут в двух репозиториях, локальном (расположен на компьютере разработчика, причем именно репозиторий с полным объемом данных, а не ссылки на отдельные версии проекта) и удаленном (лежит на удаленном сервере). Получение данных с удаленного репозитория идет через гит-хостинги Github, Google Code, GitLab и прочее.

Достаточно лишь иметь интернет-соединение, чтобы получать доступ к удаленному репозиторию. Взаимодействие просто представляет собой синхронизацию локального и удаленного репозиториев.

Любые изменения в документе и его новые версии сохраняются в локальном репозитории.

Набор всех сохраненных версий называют «деревом» проекта, и оно находится в репозитории.

Бывает прямое дерево, когда сохранения файлов выполняются последовательно, одно за другим без возврата к предыдущим вариантам.

Есть еще разветвленное дерево. Оно получается, если приходится дорабатывать старые версии, вносить в них изменения и потом сохранять.

Любые сохранения, получившиеся на ветках дерева, изначально наращиваются на один файл, называемый исходным. В процессе работы в него вносятся изменения. Система управления версиями позволяет одновременно корректировать, дополнять любую из веток дерева. В итоге некоторые из них «срастаются». Все изменения обязательно каждый раз сохраняются в последней версии файла.

Для настройки и последующей работы с системой контроля версий GIT понадобится регистрация на каком-либо из git-хостингов, к примеру на Github, Sourceforge, Google Code, GitLab, Codebase или на других.

Наибольшей популярностью из названых пользуется Github.

**Ошибки при работе с GIT**

У начинающих программистов могут возникнуть трудности при работе с системами контроля версий. Впрочем, даже опытные разработчики эти ошибки тоже допускают:

* нежелание использовать git. такое случается, когда после запуска консоли программист не может в ней разобраться. тогда можно вообще не оперировать с консолью, взяв для работы github вместо десктопной версии;
* невнимательное изучение официальной справки. а между тем, в ней много ценных сведений, помогающих существенно упростить процесс взаимодействия с репозитариями;
* несоблюдение правил работы с git. нарушение системных требований может стать причиной потери всех результатов работы над проектом. к примеру, в основную ветку нельзя вливать изменения, не прошедшие проверку;
* ошибочное использование команд системы контроля версий. запускайте команду лишь в том случае, когда точно знаете, какая операция при этом будет выполнена;
* некорректные комментарии к коммитам. практический совет: проработав несколько часов над кодом, не спешите сразу перебрасывать файлы на сервер. составьте комментарии к ним чуть позже, на «свежую голову», чтобы не наделать ошибок и не ввести в заблуждение коллег, которые будут работать с файлами после вас;
* выгрузка лишних файлов. то, что не нужно для проекта, легко удаляется либо исключается из структуры.