Привет! Этот документ – инструкция по выполнению практического задания. В нем вы найдете вспомогательные материалы по работе с Git, GitHub и DBeaver на страницах 2-11. На странице 12 начинается практическое задание. Рекомендуем сразу перейти к выполнению задания и при необходимости ознакомиться со вспомогательными материалами.

Инструкция по работе с GitHub

Термины и определения

Если в процессе работы с инструкцией вы столкнулись с непонятными или новыми словами, вернитесь к этой таблице.

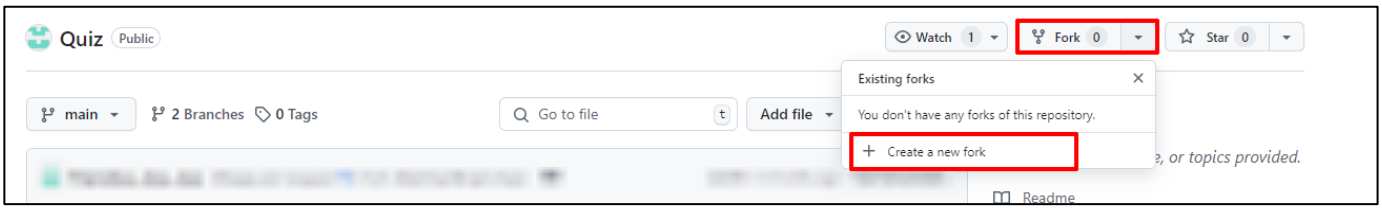
Ветка (Branch)	Параллельная версия *репозитория*. Она не влияет на главную версию, тем самым позволяя свободно работать в параллельной.
Индекс	Временное хранилище, где лежат имена файлов и их изменения, которые должны быть в следующем *коммите*. В индекс нужно явно добавлять файлы при помощи команды git add.
Коммит	Коммит хранит изменённые файлы (которые внесены в индекс), имя автора коммита и время, в которое был сделан коммит. Можно считать, что коммит является точкой сохранения состояния вашего репозитория.
Клонирование	Скачивание *репозитория* с удалённого сервера на локальный компьютер для дальнейшей работы с этим каталогом как с репозиторием при помощи команды git clone.
Локальный репозиторий	*Репозиторий*, расположенный на локальном компьютере разработчика в каталоге. Именно в нём происходит разработка и фиксация изменений, которые в дальнейшем отправляются на *удаленный репозиторий*.
Пулреквест (Pull Request)	Запрос на слияние *форка* репозитория с основным *репозиторием* на *GitHub*. Пулреквест может быть принят или отклонён владельцем оригинального репозитория.
Пуш (Push)	Отправка всех неотправленных коммитов на удалённый сервер репозитория.
Рабочий каталог	Извлеченная из базы копия определённой версии проекта.
Репозиторий	Директория проекта на платформе в интернете (удаленный) или на локальном компьютере (локальный). В директории хранится проект, и история изменений репозитория в скрытой директории «.git».
Форк (Fork)	Копия репозитория. Ее также можно рассматривать как внешнюю *ветку* для текущего репозитория. Копия вашего открытого репозитория на GitHub может быть сделана любым пользователем, после чего он может прислать изменения в ваш репозиторий оставить запрос на слияние с вашим проектом.
Git	Распределенная система управления версиями.
Git Bash	Приложение для сред Windows, эмулирующее работу командной строки Git.
GitHub	Веб-сервис для размещения репозиториев и совместной разработки проектов.

Как создать fork?

Если вы хотите внести изменения в чужой репозиторий на который у вас нет прав (*вы не являйтесь разработчиком в этом репозитории*), вы можете создать своё собственное ответвление (*fork*) этого проекта. Это означает, что GitHub создаст вашу собственную копию чужого проекта, которая будет находиться в вашем пространстве на GitHub.

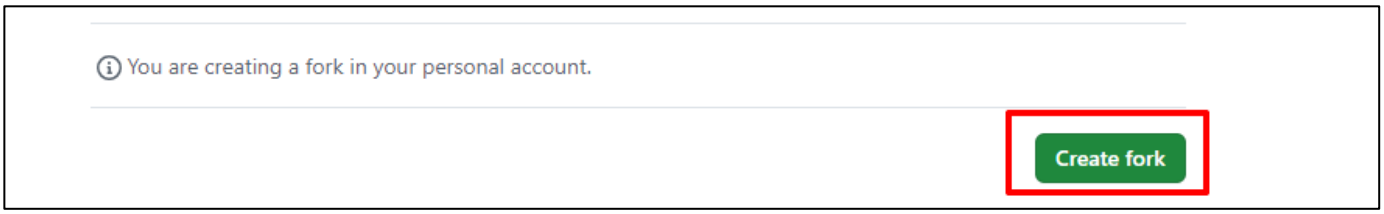
Для того чтобы создать копию (fork) чужого репозитория, необходимо выполнить следующие шаги:

1. Перейти в нужный репозиторий с проектом на GitHub в браузере.
2. Раскрыть выпадающий список «Fork» и выбрать пункт «Create a new fork».



3. На странице настроек «Fork» ввести имя нового репозитория.

4. Остальные настройки оставить по умолчанию.
5. Нажать кнопку «Create fork» внизу страницы.

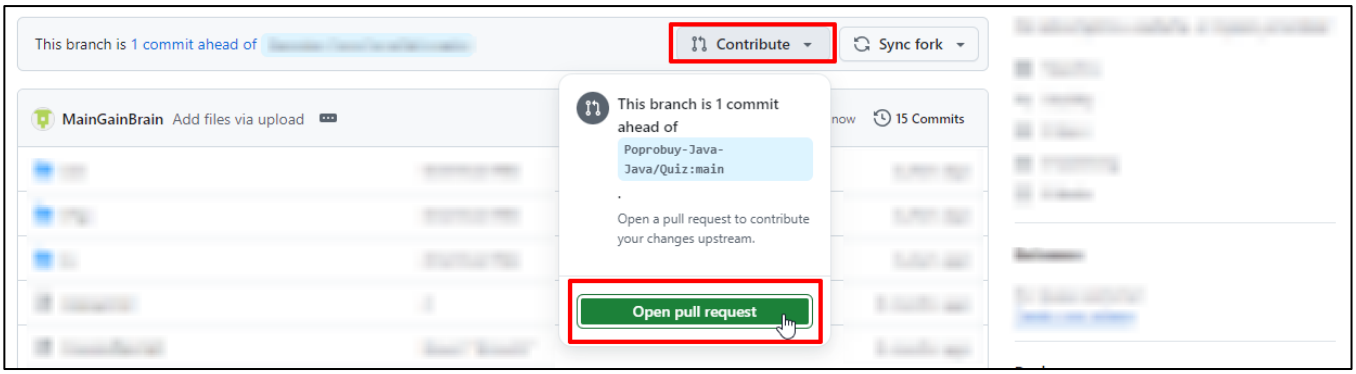


После этого в вашей библиотеке репозиториев будет лежать копия чужого репозитория, которую вы можете изменять, клонировать себе на локальный компьютер и загружать обратно на GitHub в свое пространство.

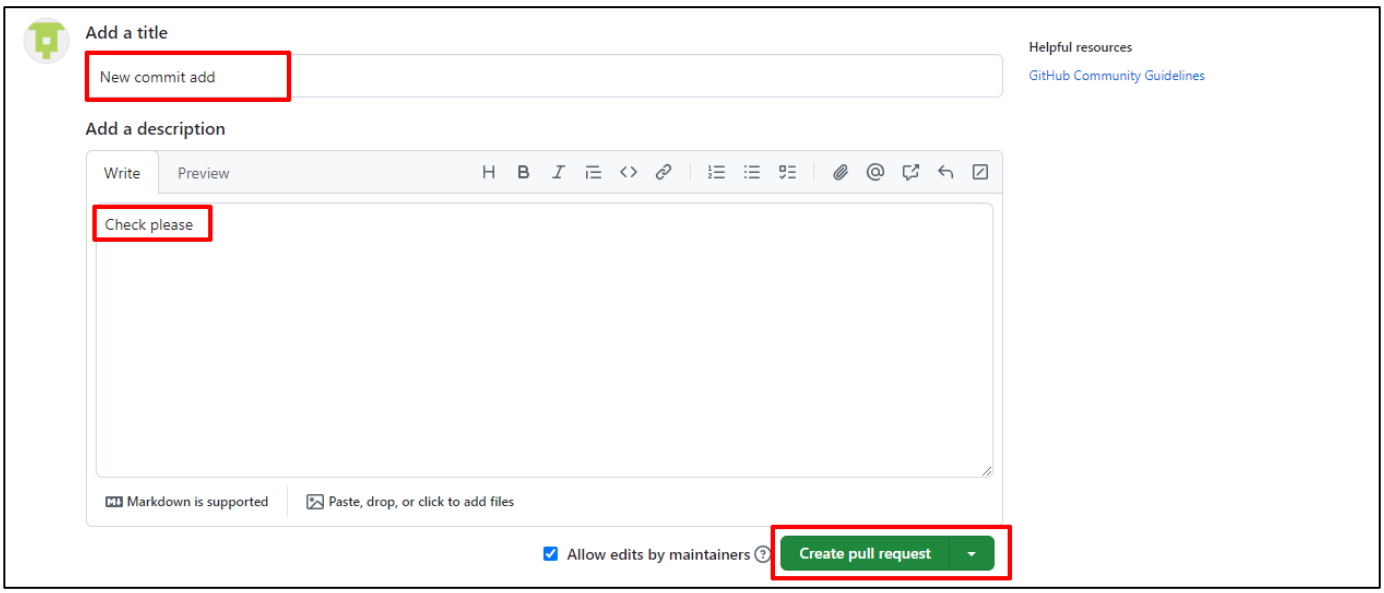
Как слить fork с репозиторием?

Для того чтобы слить свой fork и оригинальный репозиторий необходимо отправить запрос (*Pull Request*) автору оригинального репозитория. Для выполнения pull request на странице вашей копии (fork) с внесенными изменениями, необходимо:

1. Перейти в свою копию репозитория (*fork*) на GitHub в браузере**.**
2. Раскрыть выпадающий список «**Contribute**», расположенный в панели меню под зеленой кнопкой «**Code**» и Нажать на зеленую кнопку «**Open pull request**».

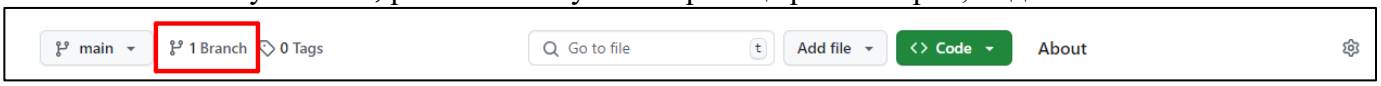


3. На открывшейся странице ввести название и описание pull request и нажать на зеленую кнопку «**Create pull request**».



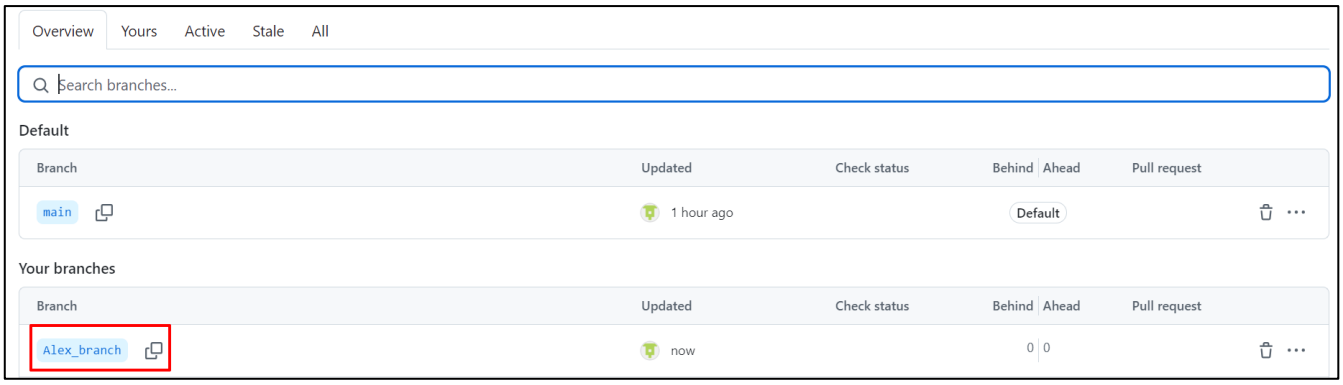
Работа с ветками на GitHub

В ваших репозиториях или «форках» вы можете создавать новые ветки проекта. Ветка является параллельной версией репозитория. Она включена в репозиторий, но не влияет на главную версию, тем самым позволяя свободно работать в параллельной.
Для работы с ветками (*создание новых веток, просмотр существующих, переход между ветками*) нажмите на кнопку **Branch**, расположенную на странице репозитория, под его названием.



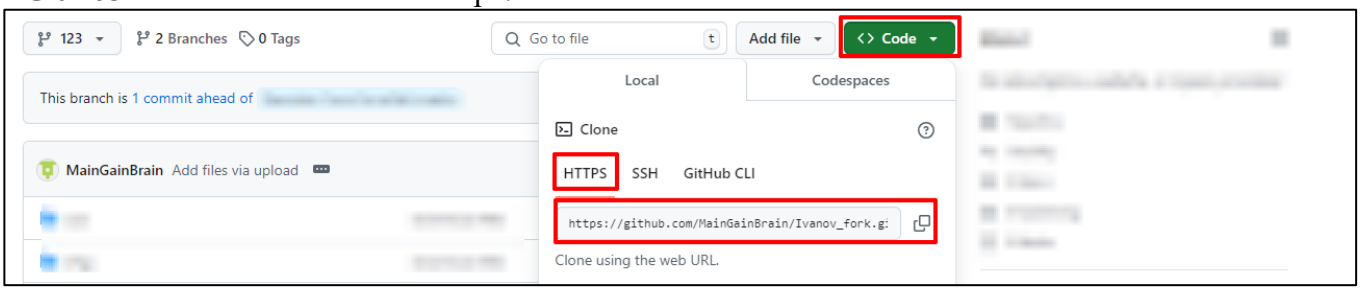
В открывшемся окне, на вкладке **Overview**, **All** или**Yours** перечислены все существующие на данный момент ветки выбранного репозитория. Создание новой ветки инициализируется кнопкой **New Branch.**

Для создания новой ветки необходимо указать имя этой ветки и нажать кнопку **Create New Branch**. Новую ветку можно будет найти в списке **All** или **Yours**. Для перехода в любую ветку, нужно нажать на ее название в любой из вкладок левой кнопкой мыши.



Ссылка на удаленный репозиторий

Для работы с репозиторием на локальном компьютере, его необходимо скачать, или клонировать из удаленного репозитория. *Ссылку для клонирования* репозитория можно получить, нажав на зеленую кнопку «**Code**» в репозитории**.** В раскрывшемся блоке на вкладке HTTPS расположена ссылка на репозиторий, которую необходимо скопировать и использовать при выполнении команды git clone в Git Bash на локальном компьютере.



Как сделать токен личного доступа?

Работа со скаченным репозиторием на локальной машине выполняется при помощи терминала GitBash. При использовании Git Bash для загрузки локального репозитория обратно в GitHub, вам потребуется доступ к удаленному репозиторию на платформе, если вы работаете не на своем компьютере без доступа к удаленному репозиторию. Для этого в GitHub реализованы токены личного доступа.

Токены личного доступа предназначены для доступа к ресурсам GitHub от вашего имени. То есть для того, чтобы залить изменения с локального репозитория на удаленный нужно использовать токен. Для создания токена, нажмите на иконку своего профиля GitHub в браузере, в правом верхнем углу страницы, и перейдите в раздел создания токена по следующему пути:

Settings (меню справа)→ **Developer Settings (**меню слева в самом низу) → **Personal Access Tokens (**меню слева) → **Tokens (classic) (**меню слева**)**

Далее нажмите кнопку (справа) «**Generate New Token»** и выберите пункт **«Generate new token (classic)».** В поле **Note** введите название токена (Необходимо для идентификации токена, не влияет на сам токен или аккаунт). В блоке «**Select scopes» нажмите на основной чекбокс repo (*чтобы* были выбраны все 5 дочерних чекбоксов). Затем внизу страницы нажмите кнопку «Generate token».**
Сам по себе токен представляет строку из цифр и чисел, которую необходимо **сохранить и запомнить** (например, записать в блокнот), так как она будет отображена пользователю всего один

раз. При закрытии окна добавления нового токена вы больше не сможете его увидеть и, если вы забыли или потеряли токен, нужно будет удалить старый и создать новый.

Добавление токена в локальный репозиторий

Чтобы добавить токен к *существующему* локальному репозиторию на компьютере, необходимо вставить его в конфигурационный файл «**config**» в локальном репозитории. Файл config расположен в скрытой папке «**.git**» локального репозитория. Для отображения скрытой папки, необходимо в проводнике во вкладке **Вид** выбрать чекбокс **Скрытые элементы**. Для того чтобы добавить токен необходимо:

1. Открыть файл «config» из директории «.git» в текстовом редакторе или блокноте
2. Найти в файле строку «url=…» с адресом удаленного репозитория в блоке [remote “origin”]. Она может выглядеть следующим образом:

```
        ignorecase = true
[remote "origin"]
        url = https://github.com/MainGainBrain/Ivanov_fork.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
        remote = origin
        merge = refs/heads/main
```

3. Вставить в строку после «https:// » название вашего аккаунта и двоеточие , например:

```
        ignorecase = true
[remote "origin"]
        url = https://ivanov:github.com/MainGainBrain/Ivanov_fork.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
        remote = origin
        merge = refs/heads/main
```

4. После двоеточия вставить токен и после него символ «@»

```
        ignorecase = true
[remote "origin"]
        url = https://ivanov:ghpp3y42121rpYKHu231z0qWX9Ls62UXr2mCPq1@github.com/MainGainBrain/Ivanov_fork.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
        remote = origin
        merge = refs/heads/main
```

5. Сохранить и закрыть конфигурационный файл.

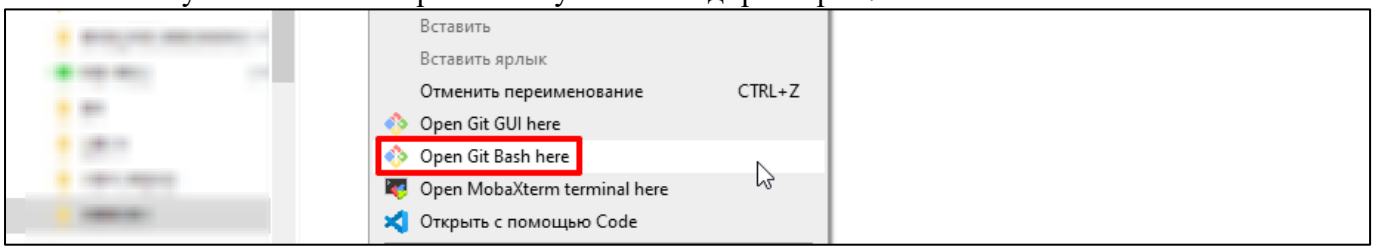
Теперь в дальнейшем, при необходимости вы сможете загрузить изменения локального репозитория через терминал Git Bash на удаленный репозиторий.

Работа с Git Bash

Git Bash представляет собой командную строку для работы с Git на операционных системах Windows. Она предоставляет удобный доступ к множеству команд Git, а также дополнительным утилитам Unix. Это значит, что в ней вы можете использовать такие команды как *cat, mkdir, touch, ls, cd, pwd* и т.д. Также в Git Bash можно вставлять скопированный текст, используя правую клавишу мыши и параметр Paste.

По аналогии с терминалом Linux систем, при выполнении команды в Git Bash, она будет выполнена в той директории, в который вы сейчас находитесь в терминале Git Bash. Git Bash можно открыть двумя способами:

1. Из меню пуск. В этом случае для работы с нужной директорией будет необходимо перейти в нее при помощи команды cd.
2. В контекстном меню, при нажатии правой кнопкой мыши на пустое пространство директории. В этом случае Git Bash откроется в нужной вам директории.



Основные команды git:

Примерный жизненный цикл удаленного репозитория при работе с ним выглядит следующим образом:
Клонирование удаленного репозитория на локальный компьютер → внесение необходимых изменений в файлы репозиторий → Локальное сохранение внесенных изменений, создание коммита → Отправка изменений на удаленный репозиторий.

Для каждого из этапов этого цикла реализована соответствующая команда, с необходимыми параметрами. Основные команды для работы с удаленными репозиториями описаны ниже.

git clone ***ссылка на удаленный репозиторий*** **-** клонировать удаленный репозиторий по указанной ссылке на свой компьютер. Репозиторий появится в той директории, где вы находитесь на момент клонирования, в терминале GitBash.

git clone ***ссылка на удаленный репозиторий*** **-b** ***имя клонируемой ветки*** **-** клонировать ветку удаленного репозитория по указанной ссылке на свой компьютер. В качестве имени ветки используется имя существующей ветки на GitHub которую вы собираетесь клонировать.

git status - посмотреть статус файлов в локальном репозитории. Выполнение данной команды выводит в терминал информацию об измененных, удаленных или добавленных файлах, которые еще не были добавлены в коммит или в индекс. Если вы не добавили файл в индекс он будет подсвечен красным цветом, если добавили – зеленым.

git add ***“имя файла”*** **-** добавить файл в индекс. После добавления указанного файла в индекс, вы сможете сделать коммит с указанным файлом.

git add . - добавить в индекс все измененные и добавленные в локальный репозиторий файлы.

git commit -m ***“Комментарий к комиту”*** **-** сделать коммит. Записывает локальные изменения в репозитории. В скобках указывается комментарий к коммиту.

git push origin ***имя-ветки*** **–** отправить зафиксированные изменения с локального клонированного репозитория в удаленный. В качестве параметра ***имя ветки*** указывается название той ветки на GitHub в удаленном репозитории, в которую планируется загрузить изменения.

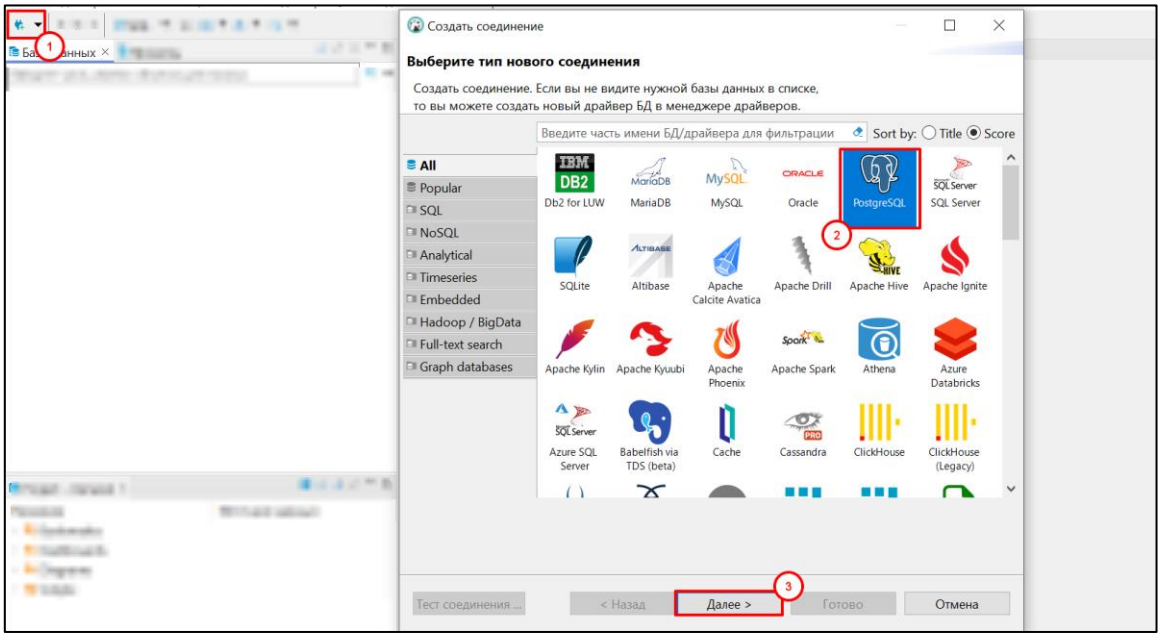
DBeaver

DBeaver — это приложение, предназначенное для управления базами данных на локальном или удаленном устройстве. Данный инструмент совместим со многими известными СУБД, среди которых — MySQL и PostgreSQL.

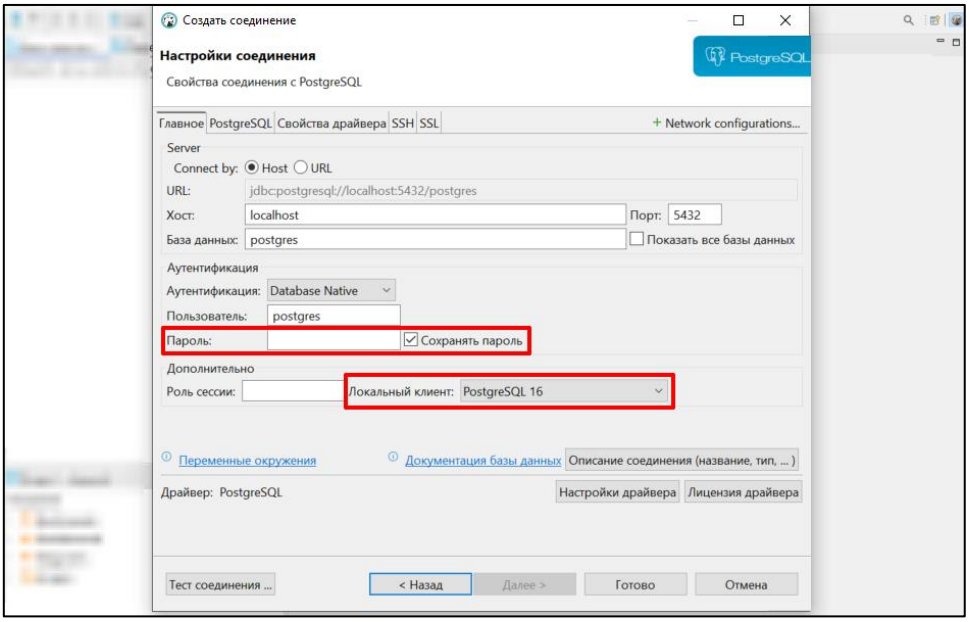
Новое соединение с базой данных

Для подключения к существующей базе данных через DBeaver необходимо выполнить несколько простых шагов.

Шаг 1. Нажать на кнопку «Новое соединение» в верхней панели окна приложения, как на скриншоте ниже. Выбрать тип БД, которую вы хотите подключить и нажать кнопку Далее.



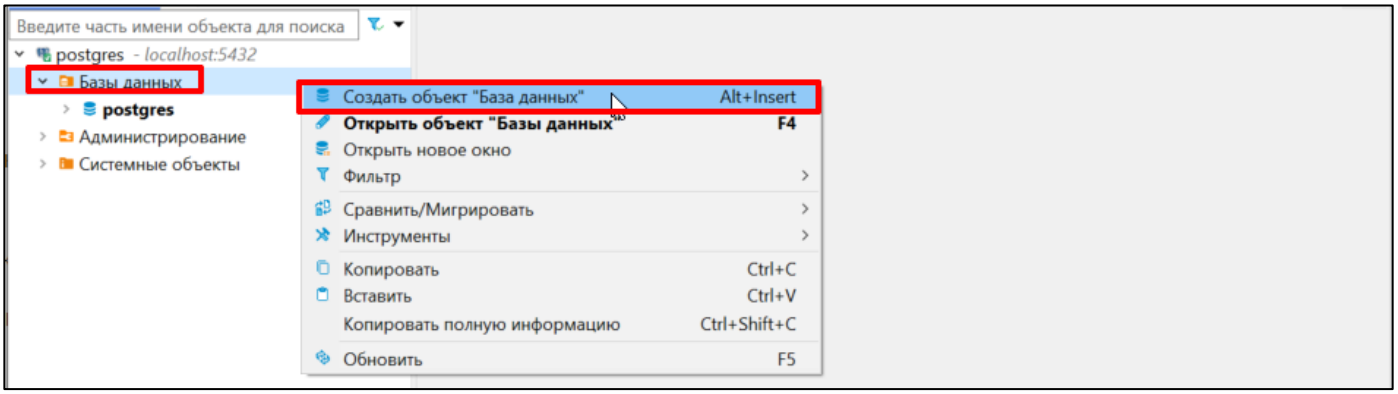
Шаг 2. В открывшемся окне необходимо ввести пароль пользователя базы данных, который был задан при установке БД, и, при необходимости выбрать версию БД, установленную на клиенте.



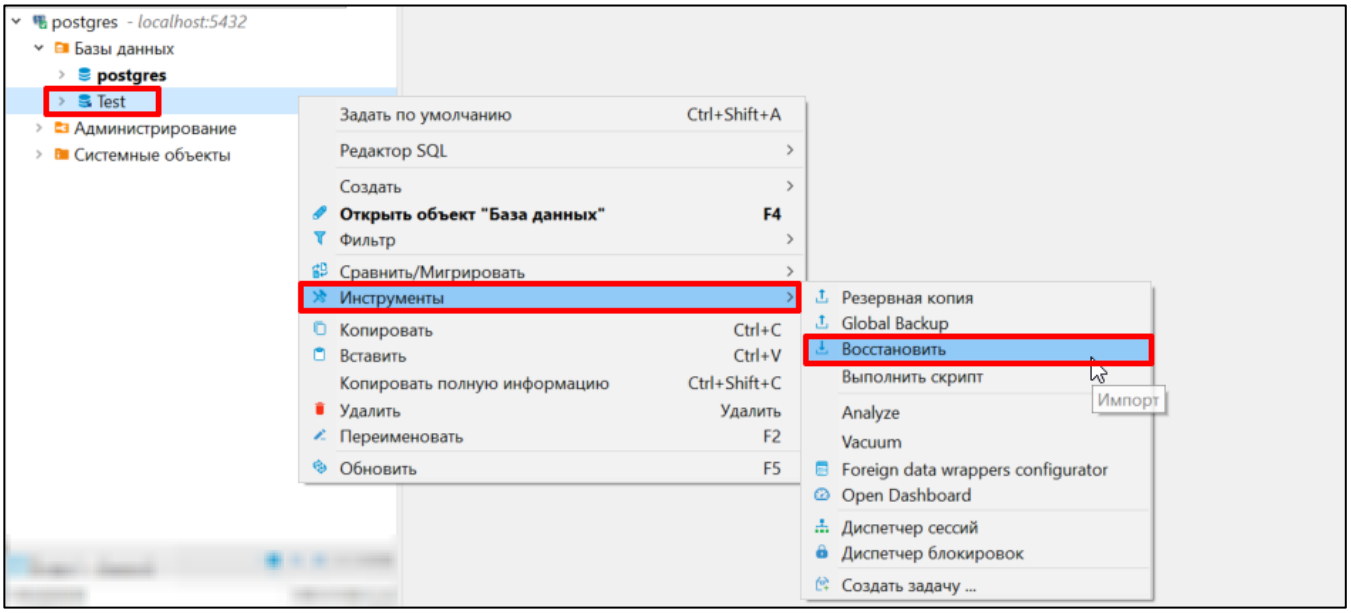
После успешного соединения, новое соединение с базой данных отобразится в окне приложения

Восстановление резервной копии БД

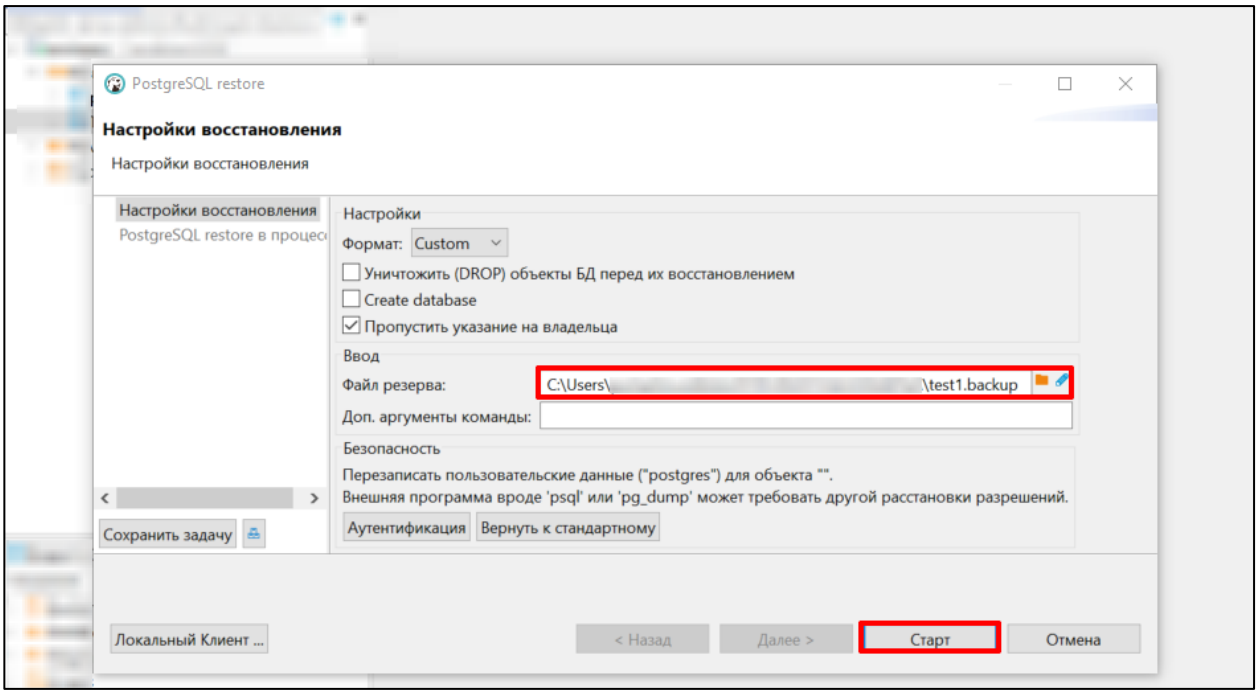
Для восстановления резервной копии базы данных необходимо сначала создать новую пустую базу данных в DBeaver. Для этого нужно нажать правой кнопкой мыши на раздел «Базы данных» и выбрать соответствующий пункт контекстного меню.



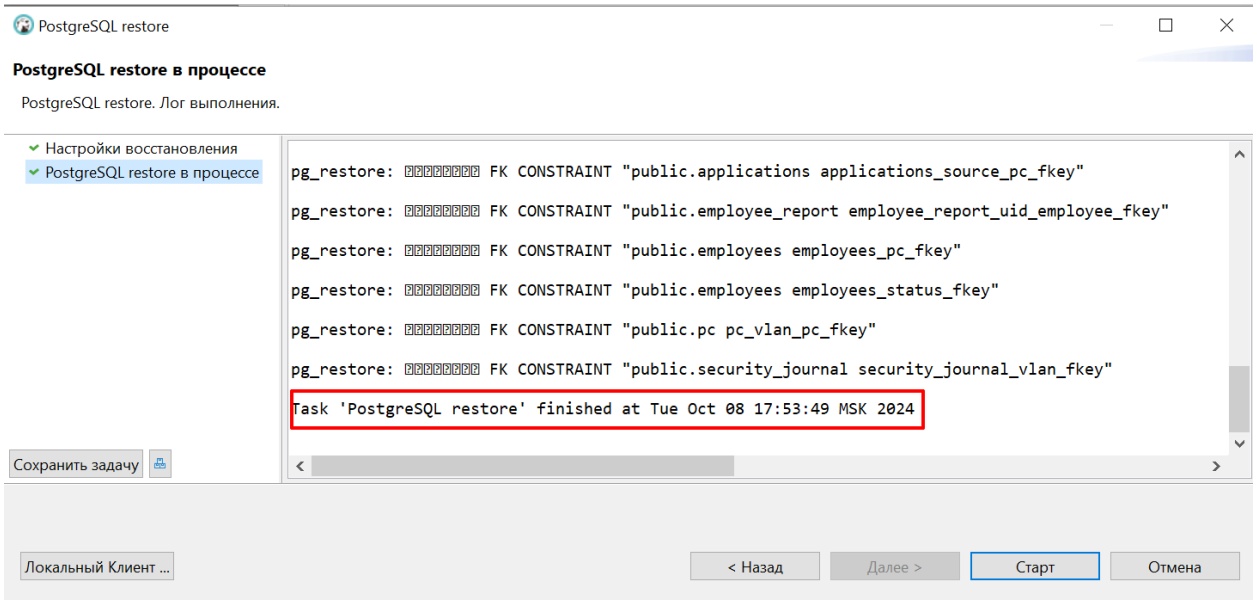
Далее в новом открывшемся окне необходимо ввести название новой базы данных и нажать кнопку «Ок». После этого в списке появится новая БД. Далее нужно восстановить данные в новую БД из файла с резервной копией. Для этого нужно нажать правой кнопкой мыши на новую БД, выбрать в контекстном меню раздел «Инструменты» и пункт «Восстановить».



В открывшемся окне необходимо указать путь до файла с резервной копией БД. И нажать кнопку старт. Остальные параметры можно оставить по умолчанию.



Запустится скрипт выполнения восстановления резервной копии. После успешного выполнения восстановления БД, закройте все дополнительные окна.



Теперь вы можете просмотреть таблицы восстановленной резервной копии БД и ознакомиться с ее схемой, выполнить запросы к БД и т.д.

Практическое задание

Для выполнения практического задания вам понадобится ваш аккаунт на GitHub и установленные на компьютер терминал Git Bash, СУБД PostgreSQL и приложение DBeaver.

Задание 1. Работа с GitHub

Шаг 1. Войдите в свой аккаунт GitHub в браузере.
Шаг 2. Перейдите в следующий репозиторий по ссылке:

https://github.com/MainGainBrain/task2

Шаг 3. Создайте fork удаленного репозитория. В качестве названия «форка» укажите свою фамилию.
Шаг 4. Создайте новую ветку в сделанном «форке». В качестве названия ветки используйте слово «branch» и свою фамилию. Перейдите в созданную только что ветку.
Шаг 5. При необходимости, если вы используете не свой компьютер или не работали до этого с Git Bash, для загрузки файлов в удаленный репозиторий, создайте токен личного доступа для своего аккаунта. Не забудьте его скопировать и сохранить, так как отображение токена будет доступно только один раз.

Задание 2. Работа с Git Bash

Шаг 6. Создайте новую папку на рабочем столе. Назовите ее по своему усмотрению.
Шаг 7. Откройте терминал Git Bash в созданной на шаге 6 папке.
Шаг 8. С помощью Git Bash клонируйте ветку вашего репозитория, созданную в шаге 4, к себе в папку на локальный компьютер.
Шаг 9. Откройте новое окно терминала Git Bash в папке скачанной ветки на локальном компьютере**.** Если вы правильно клонировали ветку, то в GitBash будет отображено название выбранной ветки голубым цветом. Если название ветки в терминале не соответствует названию вашей ветки в GitHub или отсутствует, значит вы находитесь не в той директории или скачали не ту ветку.
Шаг 10. При необходимости добавьте созданный на шаге 5 токен в конфигурационный файл «*config»* в локальной версии репозитория.
Шаг 12. При помощи проводника, найдите в папке файл «Практическое задание SQL.pdf». В файле представлено третье практическое задание по работе с базами данных. Откройте файл и выполните задание. Только после успешного выполнения третьего практического задания перейдите к шагу 13 в этом файле.
Шаг 13. Добавьте ответ на третье практическое задание в папку с репозиторием в формате «.txt» или «.sql».
Шаг 14. Перейдите в терминал Git Bash и добавьте новые файлы с ответом в индекс и сделайте коммит. Не забудьте закрыть все открытые окна файлов перед выполнением коммита, чтобы процесс выполнился успешно. Если вдруг при создании коммита в Git Bash будет выведено сообщение о необходимости настроить имя и почту пользователя, выполните в терминал следующие 2 команды, заменив «Your Name» на логин пользователя GitHub, а вместо «your@example.com» почту, которая была указана при регистрации на GitHub.

git config --global user.name “Your Name”
git config --global user.email your@example.com

Затем повторно сделайте коммит ветки локального репозитория.

Шаг 15. Загрузите ветку обратно в удаленный репозиторий GitHub через терминал Git Bash.
Шаг 16. Зайдите на GitHub в браузере и проверьте что все изменения загрузились.
Шаг 17. Выполните пулл реквест вашей ветки с загруженными изменениями с тем репозиторием из шага 2 с которого была сделана копия.
Шаг 18. Оповестите сотрудника о завершении выполнения практического задания.