

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО РАБОТЕ №2.11**  
**дисциплины «Основы кроссплатформенного программирования»**

Выполнил:

Кондратенко Даниил Витальевич

1 курс, группа ИТС-б-о-22-1,

11.03.02 «Инфокоммуникационные  
технологии и системы связи»,

направленность (профиль)

«Инфокоммуникационные системы и  
сети», очная форма обучения

---

(подпись)

Руководитель практики:

Воронкин Р.А., канд. тех. наук, доцент,  
доцент кафедры инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

*Тема:* замыкания в языке Python.

*Цель работы:* приобретение навыков по работе с замыканиями при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

Задание 1.

Изучил теоретический материал работы, создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами.

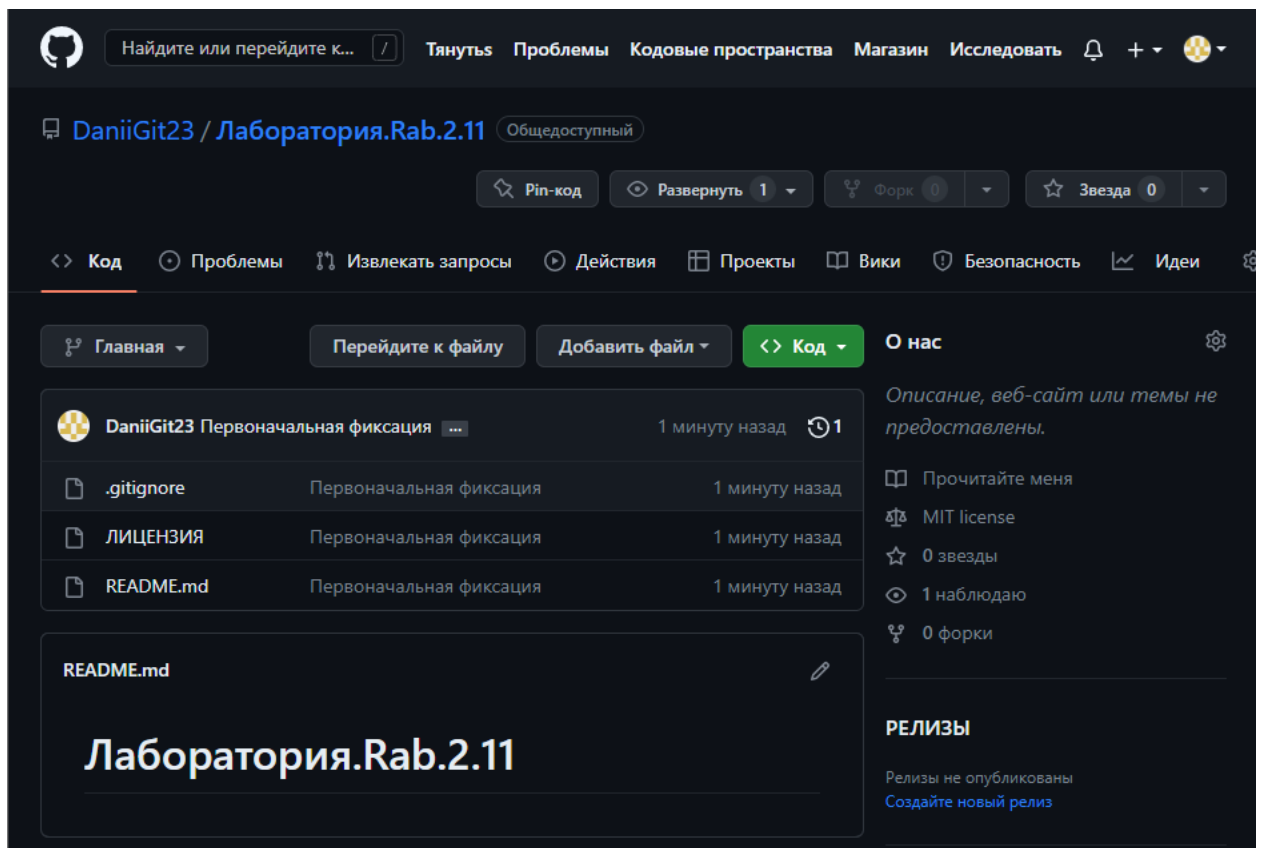


Рисунок 1 – Новый репозиторий

## Задание 2.

Проклонировал свой репозиторий на свой компьютер.

Организовал свой репозиторий в соответствии с моделью ветвления git-flow, появилась новая ветка develop.

```
C:\Users\HUAWEI>git clone https://github.com/DaniiGit23/Lab.Rab.2.11.git
Cloning into 'Lab.Rab.2.11'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\Users\HUAWEI>cd C:\Users\HUAWEI\Lab.Rab.2.11

C:\Users\HUAWEI\Lab.Rab.2.11>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .idea/

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\HUAWEI\Lab.Rab.2.11>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] t
Hooks and filters directory? [C:/Users/HUAWEI/Lab.Rab.2.11/.git/hooks]
```

Рисунок 2 – Клонирование и модель ветвления git-flow

Реализовывал примеры и индивидуальные задания на основе ветки develop, без создания дополнительной ветки feature/(название ветки) по указанию преподавателя.

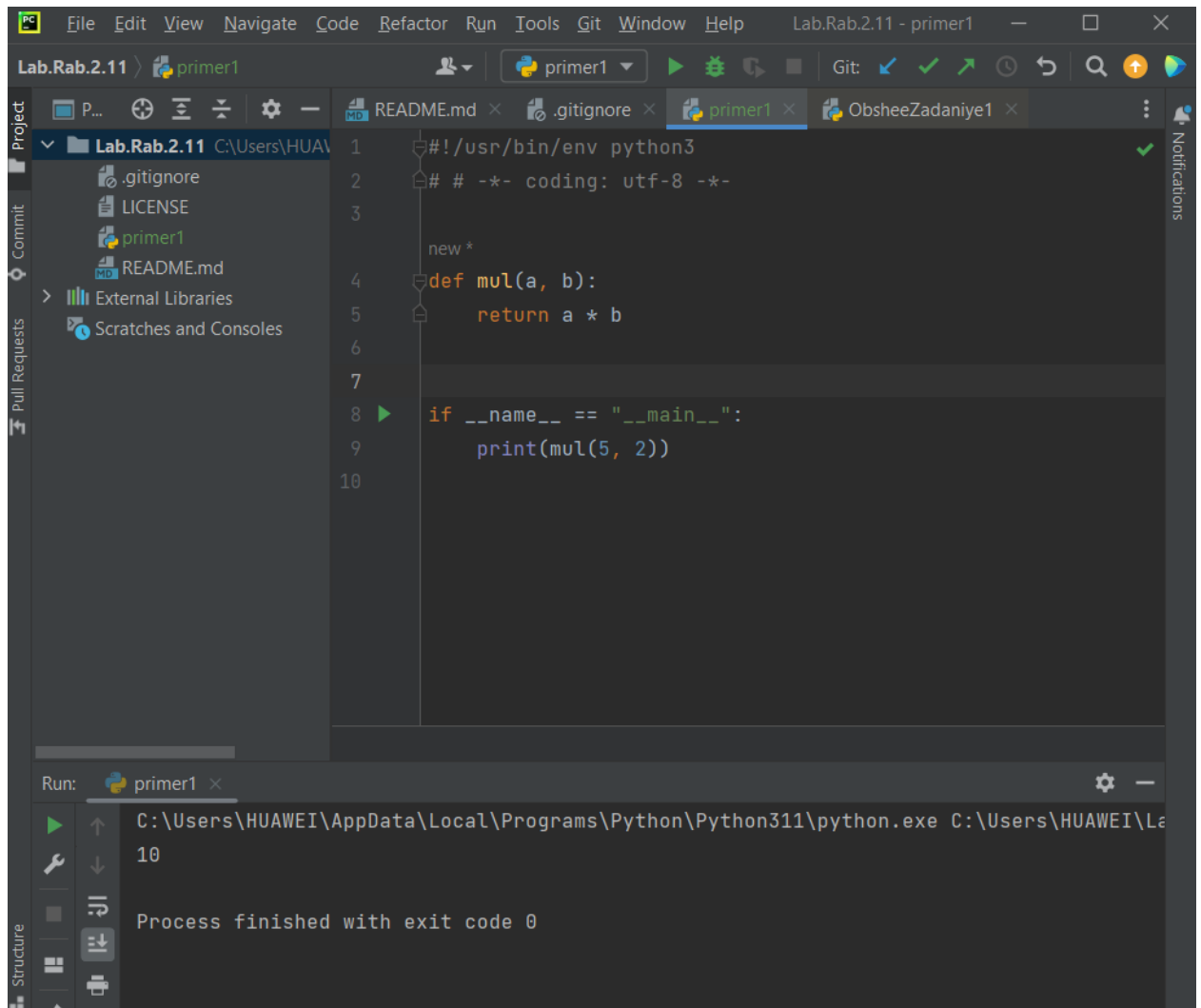
### Задание 3.

Создал проект PyCharm в папке репозитория.

Работа с примером №1.

Добавил новый файл *primer1.py*

Условие примера: функция `mul()` умножает два числа и возвращает полученный результат. Если мы ходим на базе нее решить задачу: “умножить число на пять”, то в самом простом случае, можно вызывать `mul()`, передавая в качестве первого аргумента пятерку.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def mul(a, b):
5      return a * b
6
7
8  if __name__ == "__main__":
9      print(mul(5, 2))
10
```

Run: primer1

C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\La  
10

Process finished with exit code 0

Рисунок 3 – Выполнение примера и его результат

Зафиксировал данные изменения.

Задание 4.

Создал проект PyCharm в папке репозитория.

Работа с примером №2.

Добавил новый файл *primer2.py*

Условие примера: на самом деле мы можем создать новую функцию, которая будет вызывать `mul()`, с пятеркой и ещё одним числом, которое она будет получать в качестве своего единственного аргумента.

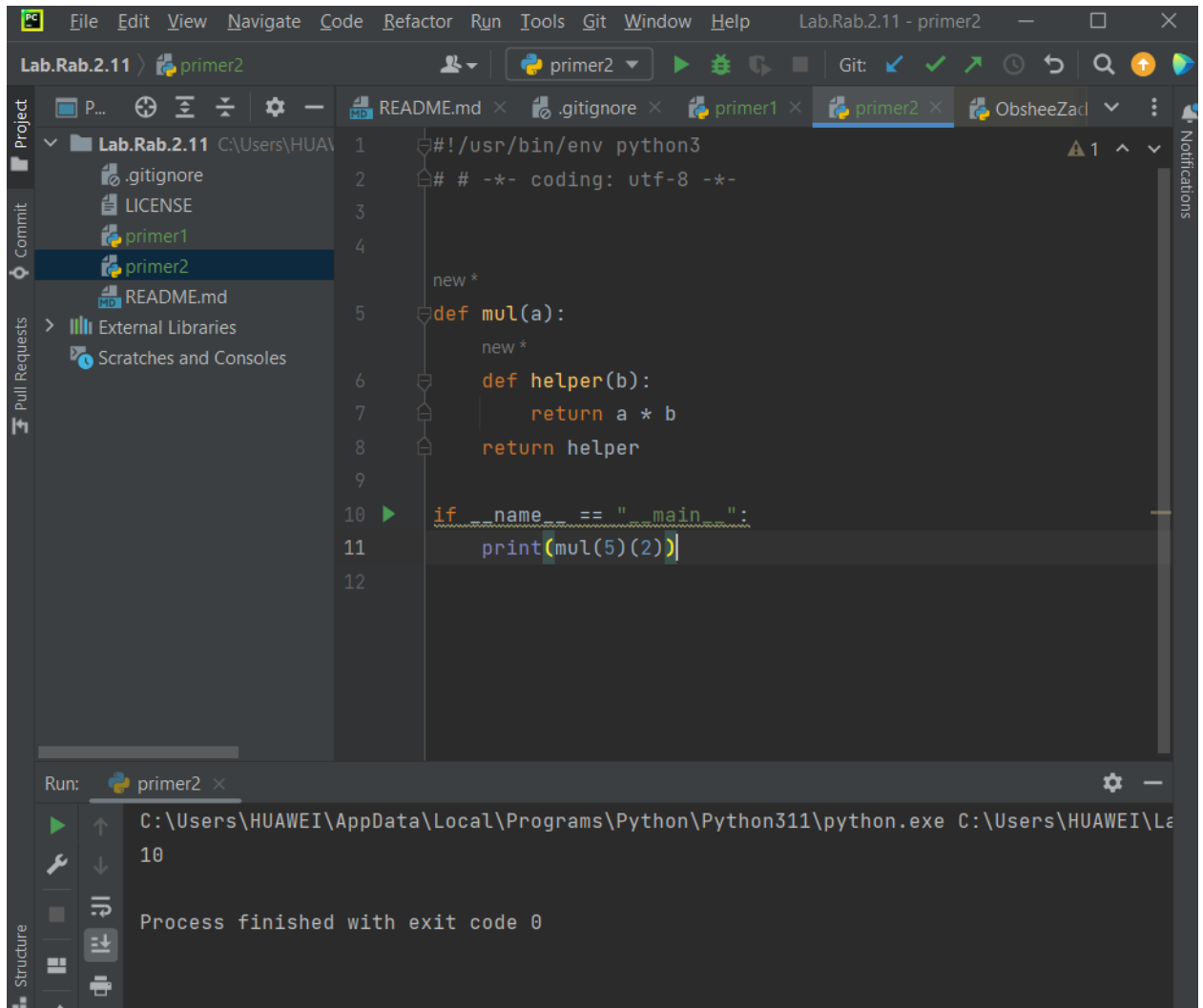


Рисунок 4 – Выполнение примера и его результат

Зафиксировал данные изменения.

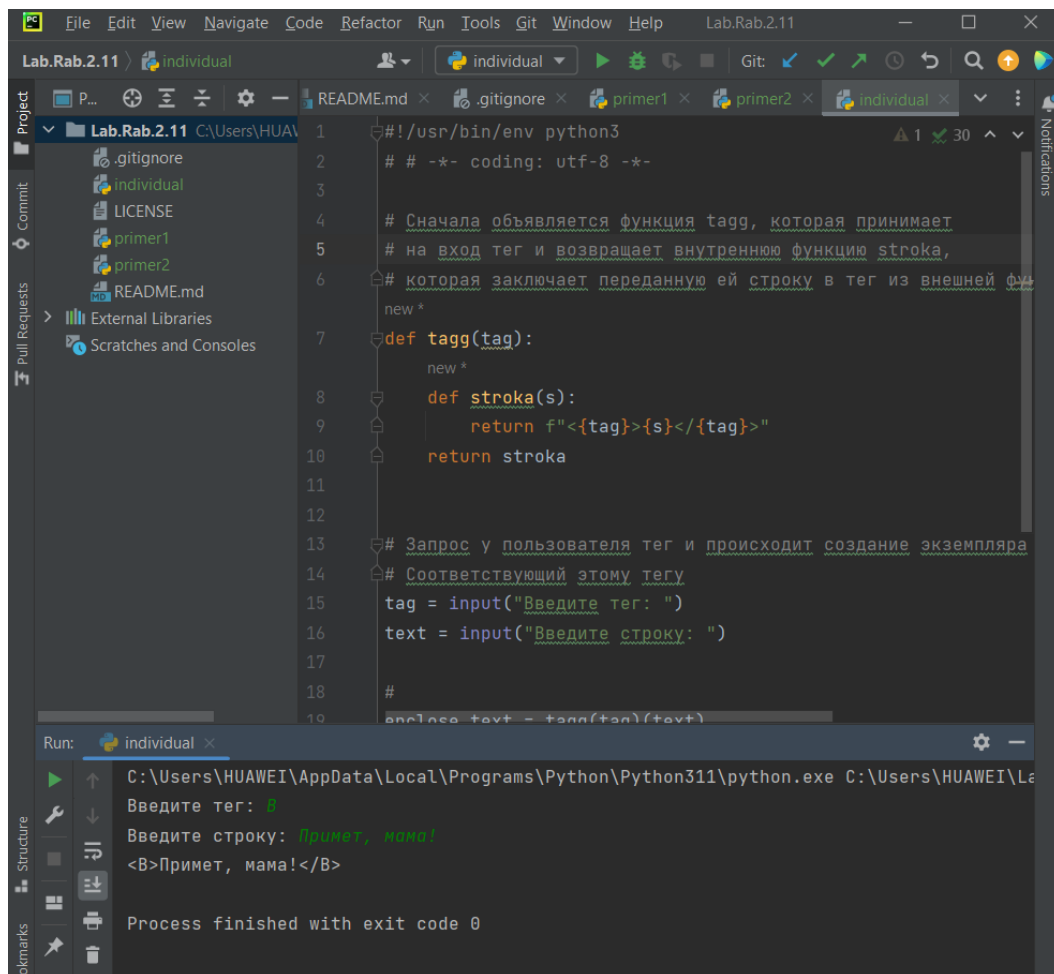
## Задание 5.

Выполнение индивидуального задания.

Вариант в соответствии со списком – 2. (по списку 12, т.к. заданий всего 10 значит вариант 2)

Создал новый файл под названием *individual.py*

Условие индивидуального задания: Используя замыкания функций, объявите внутреннюю функцию, которая заключает строку *s* (*s* – строка, параметр внутренней функции) в произвольный тег, содержащийся в переменной *tag* – параметре внешней функции. Далее, на вход программы поступает две строки: первая с тегом, вторая с некоторым содержимым. Вторую строку нужно поместить в тег из первой строки с помощью реализованного замыкания. Результат выведите на экран.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Сначала объявляется функция tagg, которая принимает
5  # на вход тег и возвращает внутреннюю функцию stroka,
6  # которая заключает переданную ей строку в тег из внешней функции
7  def tagg(tag):
8      def stroka(s):
9          return f"<{tag}>{s}</{tag}>"
10         return stroka
11
12
13 # Запрос у пользователя тег и происходит создание экземпляра
14 # Соответствующий этому тегу
15 tag = input("Введите тег: ")
16 text = input("Введите строку: ")
17
18 #
19 enclose_text = tagg(tag)(text)
```

Run: individual ×

C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\La

Введите тег: <B>

Введите строку: Привет, мама!

<B>Примет, мама!</B>

Process finished with exit code 0

Рисунок 5 – Выполнение примера и его результат

Зафиксировал данные изменения.

## Задание 6.

Слил ветку develop с веткой main, для начала перешел на ветку main (*git checkout main*), далее с помощью команды *git merge develop* слил ветку с основной и отправил на удаленный сервер.

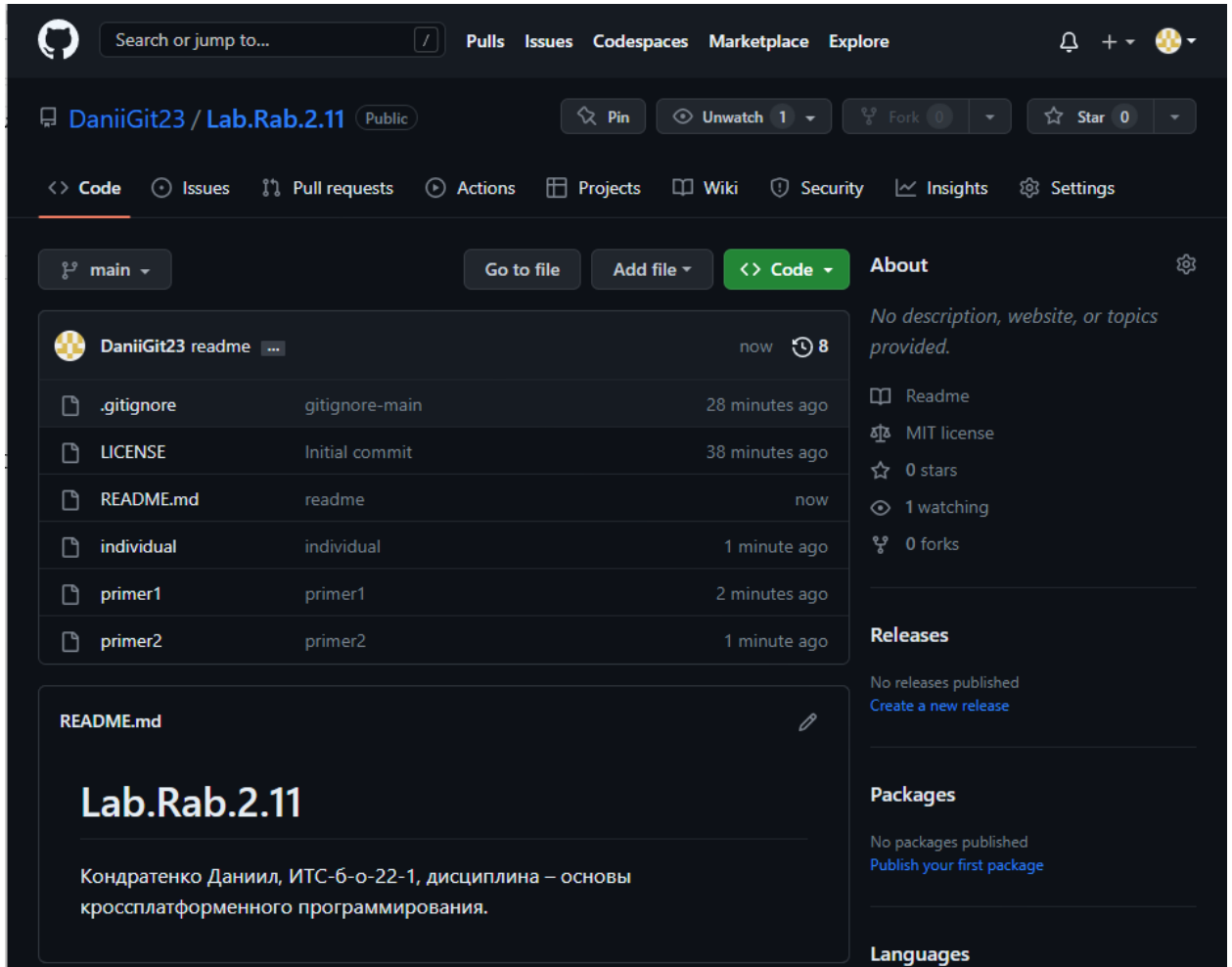


Рисунок 6 –Удаленный сервер

Ссылка на репозиторий: <https://github.com/DaniiGit23/Lab.Rab.2.11.git>

Ответы на контрольные вопросы:

1) Что такое замыкание?

Замыкание - это функция, которая запоминает значение из своего лексического окружения даже в том случае, если это значение изменяется или удаляется. При использовании замыкания функция сохраняет доступ к переменным из внешней (замкнутой) области видимости, даже если эта

область уже не существует в памяти. Это позволяет создавать более гибкие и мощные функции в Python.

## 2) Как реализованы замыкания в языке программирования Python?

В Python замыкание реализуется через вложенные функции. Внутренняя функция замыкает в себе переменные из внешней функции, сохраняя их состояние и доступ к ним при вызове внутренней функции. В качестве возвращаемого значения внешняя функция должна возвращать ссылку на внутреннюю функцию. Это позволяет сохранять контекст вызова, чтобы его можно было использовать при последующих вызовах внутренней функции.

## 3) Что подразумевает под собой область видимости Local?

Local - это область видимости, которая соответствует блоку внутри определения функции. Переменные, объявленные внутри функции или в блоке, возможно, определенной функцией, называются локальными переменными и видны только в пределах этой функции или этого блока. Это означает, что локальные переменные могут быть доступны только в той функции, в которой они были определены.

## 4) Что подразумевает под собой область видимости Enclosing?

Область видимости Enclosing (вложенная область видимости) означает, что вложенная функция имеет доступ к переменным, определенным внутри внешней функции. Эти переменные могут быть либо прочитаны, либо изменены локально во вложенной функции, но не могут быть удалены или изменены внешней функцией.

## 5) Что подразумевает под собой область видимости Global?

Область видимости Global предполагает, что переменная объявлена за пределами функции или класса и доступна везде.

## 6) Что подразумевает под собой область видимости Built-in?

Область видимости Built-in в Python относится к встроенным именам, которые автоматически включены в консоль Python. В ней находятся все стандартные имена модулей и функций, например: `print ()`, `len ()`, `str ()`, `list ()`.

## 7) Как использовать замыкания в языке программирования Python?



В языке программирования Python замыкания могут быть использованы для сохранения состояния функции и передачи его между вызовами функции. Для создания замыкания в Python необходимо создать функцию внутри другой функции, которая будет захватывать переменные из внешней функции.

8) Как замыкания могут быть использованы для построения иерархических данных?

Замыкания могут быть использованы для построения иерархических данных в языке программирования Python, например, для создания древовидных структур данных, таких как деревья решений.

Один из способов использования замыканий для построения иерархических данных - это создание функций-генераторов.

*Вывод:* в ходе данной лабораторной работы я приобрел навыки по работе с замыканиями при написании программ с помощью языка программирования Python версии 3.x.