Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 5

Название: **Основы асинхронного программирования на Golang**

Дисциплина: **Языки Интернет-программирования**

Студент	ИУ6-31Б		О.И.Ельничных
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2024

Цель работы

Изучение основ асинхронного программирования с использованием языка Golang. В рамках данной лабораторной работы предлагается продолжить изучение Golang и познакомиться с продвинутыми конструкциями языка.

Ход работы

Делаем fork репозитория (Рисунок 1).

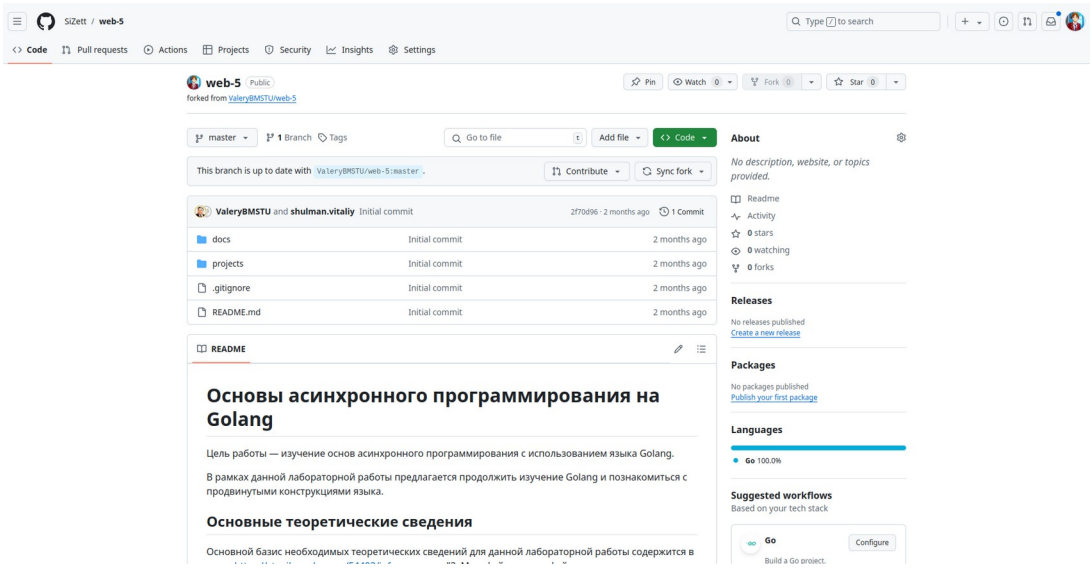

Рисунок 2

Код задания calculator:

```
package main

import "fmt"

func calculator(firstChan <-chan int, secondChan <-chan int, stopChan <-chan struct{}) <-chan int {
resultChan := make(chan int)

go func() {
defer close(resultChan)

for {
select {
case firstValue, ok := <-firstChan:
if !ok {
return
}
resultChan <- firstValue * firstValue
return
```

```
case secondValue, ok := <-secondChan:
if !ok {
return
}
resultChan <- secondValue * 3
return

case <-stopChan:
return
}
}
}()

return resultChan
}

func main() {
firstChan := make(chan int)
secondChan := make(chan int)
stopChan := make(chan struct{})

go func() {
firstChan <- 5
}()

go func() {
secondChan <- 10
}()

go func() {
stopChan <- struct{}{}
}()

resultChan := calculator(firstChan, secondChan, stopChan)

for result := range resultChan {
fmt.Println(result)
}
close(stopChan)
}
```

Код задания pipeline:

```
package main

import (
"fmt"
"sync"
)

func work() {
fmt.Println("Работаю!")
}
```

```
func main() {
wg := new(sync.WaitGroup)
for i := 0; i < 10; i++ {
wg.Add(1)
go func() {
defer wg.Done()
work()
}()
}

wg.Wait()
}
```

Код задания work:

```
package main

import (
"fmt"
"time"
// "sync"
)

func work() {
time.Sleep(time.Millisecond * 50)
fmt.Println("done")
}

func removeDuplicates(inputStream <-chan string, outputStream chan<- string) {
defer close(outputStream)

prevValue := <-inputStream
outputStream <- prevValue

for value := range inputStream {
if prevValue == value {
continue
}

outputStream <- value
prevValue = value
}
}

func main() {
inputStream := make(chan string)
outputStream := make(chan string)

go func() {
defer close(inputStream)
inputStream <- "Hello"
inputStream <- "World"
inputStream <- "World"
```

```
inputStream <- "!"
}()

go removeDuplicates(inputStream, outputStream)

for value := range outputStream {
fmt.Println(value)
}
}
```

Заключение

При выполнении заданий лабораторной работы №5 мы познакомились с основами асинхронного программирования с использованием языка Golang и выполнили три задания с разными условиями.