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).

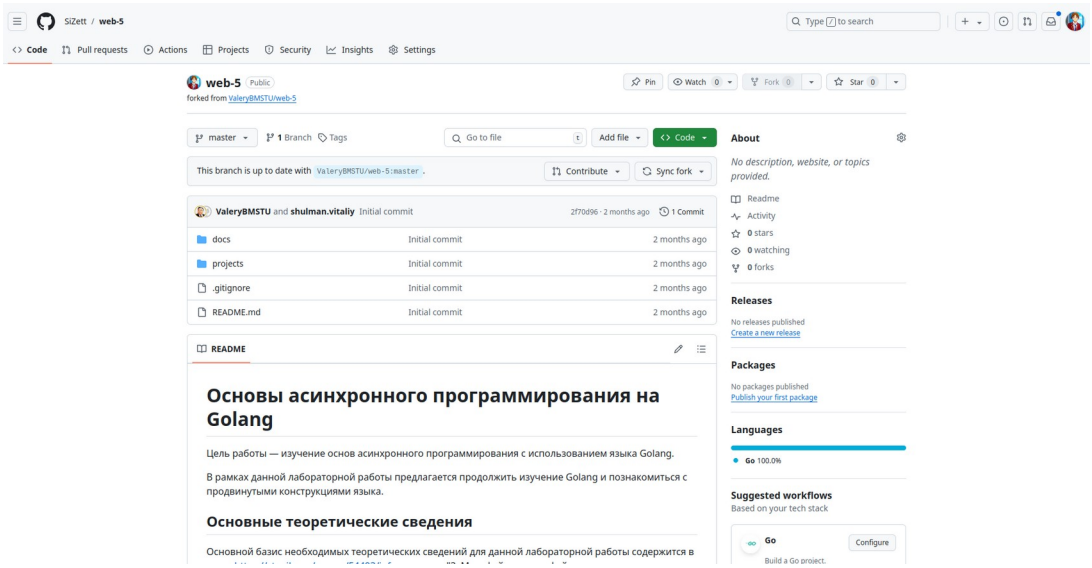

Рисунок 1

Код задания calculator:

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

 go func() {
  defer close(resultChan) // Закрываем выходной канал при выходе из goroutine

  for {
   select {
   case firstValue, ok := <-firstChan:
    if !ok {
     return
    }
    resultChan <- firstValue * firstValue
    return // Выходим из goroutine после первой обработки

   case secondValue, ok := <-secondChan:
    if !ok {
     return
```

```
    }
    resultChan <- secondValue * 3
    return // Выходим из goroutine после первой обработки

   case <-stopChan:
    return
   }
  }
 }()

 return resultChan
}
```

Код задания pipeline:

```
        wg := new(sync.WaitGroup)

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

wg.Wait()
```

Код задания work:

```
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
        }
}
```

Заключение

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