Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Санкт-Петербургский Государственный Электротехнический
Университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»

Факультет компьютерных технологий и информатики
Кафедра автоматики и процессов управления

ОТЧЕТ

по лабораторной работе №5
по дисциплине «СМиСПИС»

« Основы разработки Web-сервисов »

Студент гр. 5371 _____ Мартынов М.

Студентка гр. 5371 _____ Козлова С.

Студент гр. 5371 _____ Аверкиев В.

Преподаватель _____ Кораблев Ю.А.

Санкт-Петербург
2020

1. Цель работы

Цель работы: изучить основы разработки Web-сервисов.

2. Задание на лабораторную работу №5. Вариант №3.

Создать клиент-серверное Web-приложение по конвертации метрических и британских единиц длины.

3. Выполнение лабораторной работы

3.1 Выбор технологий

Для разработки серверной части используем Spring Boot, язык программирования Kotlin с компиляцией под JVM.

Spring Boot позволяет легко создавать автономные, производственные приложения на основе Spring, которые вы можете "просто запустить". С помощью платформы Spring и сторонних библиотек, поэтому можно начать работу с минимальной суетой. Большинство приложений Spring Boot требуют минимальной конфигурации.

Будем использовать библиотеку spring-boot-starter-web из огромного множества библиотек проекта Spring Boot. Данная библиотека включает в себя встроенный сервлет-контейнер Tomcat, что позволит быстро и просто запускать наше приложение.

Для разработки Web-клиента используем Vue JS – фрейворк для разработки Web-клиентов на языке JavaScript.

3.2 Разработка серверой части

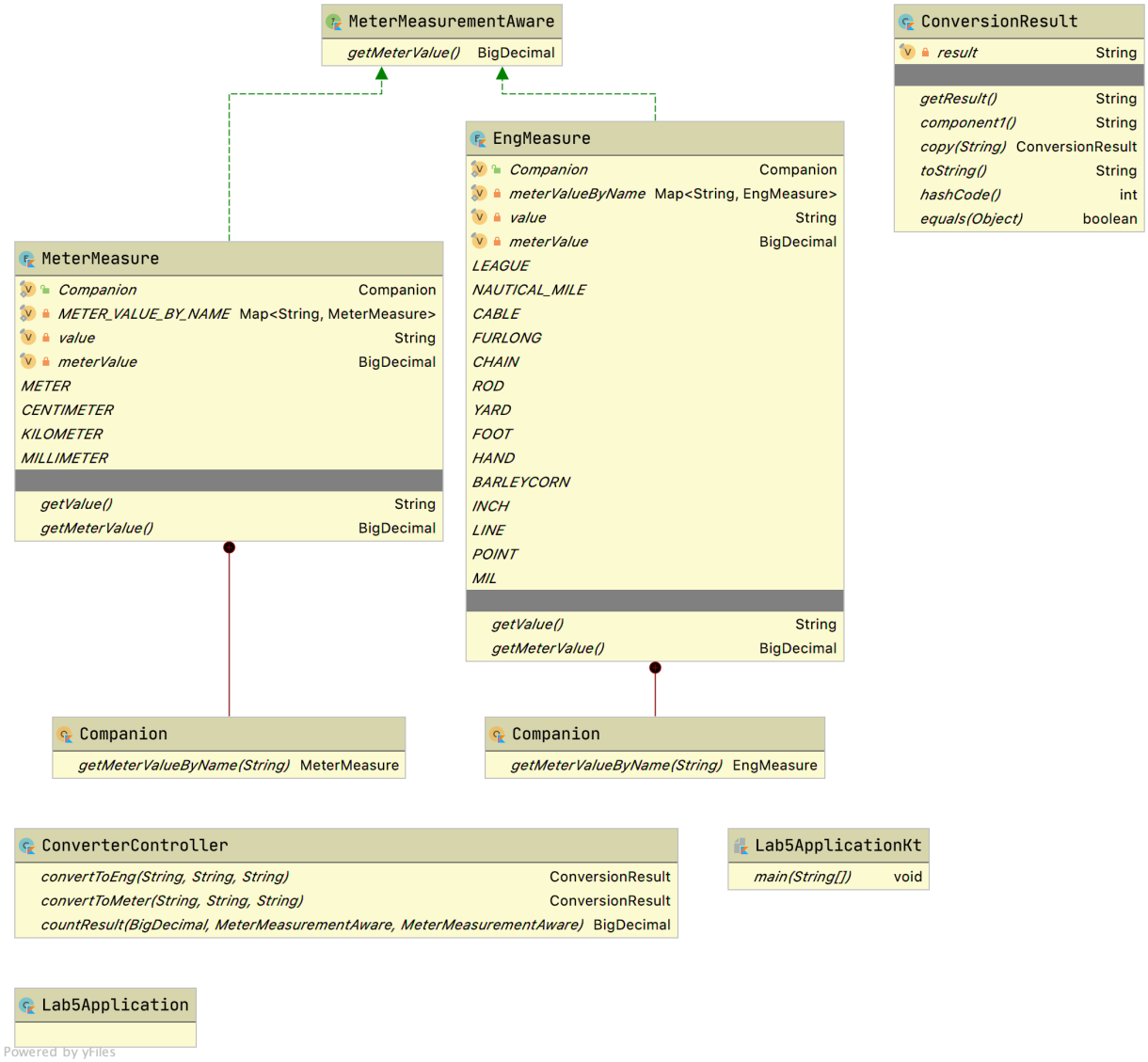
Для приема запросов по HTTP разработано следующее API через GET запросы:

1. */convert/to/eng?*
   inputValue=V1&inputMeasurement=V2&outputMeasurement=V3
   По данному API происходит конвертация значения V1, измеряемого в британской мере длины V2 в метрическую величину размерности V3.
2. */convert/to/meter?*
   inputValue=V1&inputMeasurement=V2&outputMeasurement=V3
   По данному API происходит конвертация значения V1, измеряемого в метрической системе длины V2 в английскую величину длины размерности V3.

Ответ по любому из этих запросов возвращает преобразованное занчение в формате JSON:

```
{“result”: “500.0000”}
```

3.3 UML – диаграмма разработанных классов



Ниже в таблице находится описание классов.

Название класса	Описание
ConverterController	Бизнес логика по конвертации величин из метрической системы в английскую и обратно.
MeterMeasure	Константы метрических единиц.
EngMeasure	Константы английских единиц.
ConvertionResult	Транспортный объект ответа на запрос для десериализации в JSON.

3.4 Разработка клиентской части

Код клиента, написанный на JavaScript с использованием фреймворка Vue JS и набора готовых CSS стилей Bootstrap. Состоит из двух частей template и код на JavaScript.

<template> часть

```
<template>
  <div id="greeting" class="converter">
    <h3 class="headerStyle">Конвертер мер длины</h3>
    <div class="oneRaw">
      <div class="inputStyle">
        <b-form-input type="number" placeholder="0,0" v-model="inputEng"/>
      </div>
      <div class="dropdownStyle">
        <vue-dropdown :config="configLeftEng"
@setSelectedOption="configLeftEngFunction($event)"></vue-dropdown>
      </div>
      <div class="pEqual">
        <p> = </p>
      </div>
      <div class="inputStyle">
        <b-form-input type="number" placeholder="0,0" v-model="outputMetr"
disabled/>
      </div>
      <div class="dropdownStyle">
        <vue-dropdown :config="configRightMetr"
                @setSelectedOption="configRightMetrFunction($event)"></vue-
dropdown>
      </div>
    </div>
    <div class="oneRaw">
      <div class="inputStyle">
        <b-form-input type="number" placeholder="0,0" v-model="inputMetr"/>
      </div>
      <div class="dropdownStyle">
        <vue-dropdown :config="configLeftMetr"
                @setSelectedOption="configLeftMetrFunction($event)"></vue-
dropdown>
      </div>
      <div class="pEqual">
        <p> = </p>
      </div>
      <div class="inputStyle">
        <b-form-input type="number" placeholder="0,0" v-model="outputEng"
disabled/>
      </div>
      <div class="dropdownStyle">
        <vue-dropdown :config="configRightEng"
                @setSelectedOption="configRightEngFunction($event)"></vue-
```

```
dropdown>
        </div>
      </div>
      <div class="buttonStyle">
        <b-button v-on:click="convert" variant="primary">Конвертировать</b-button>
      </div>
    </div>
</template>
```

JavaScript код клиента

```
<script>
  import axios from 'axios'
  import VueDropdown from 'vue-dynamic-dropdown';

  export default {
    data() {
      return {
        inputEng: '0,0',
        outputEng: '0,0',
        inputMetr: '0,0',
        outputMetr: '0,0',
        configLeftEng: {
          placeholder: "футов",
          options: [
            {
              value: "лиг"
            },
            {
              value: "морских миль"
            },
            {
              value: "кабельтов"
            },
            {
              value: "фурлонгов"
            },
            {
              value: "чейнов"
            },
            {
              value: "родов"
            },
            {
              value: "ярдов"
            },
            {
              value: "футов"
            },
            {
              value: "хэндов"
            },
            {
```

```
            value: "барликорнов"
          },
          {
            value: "дюймов"
          },
          {
            value: "линий"
          },
          {
            value: "точек"
          },
          {
            value: "мил"
          }
        ],
        backgroundColor: "gray",
        hoverBackgroundColor: "gray",
        border: "black"
      },
      configRightEng: {
        placeholder: "футов",
        options: [
          {
            value: "лиг"
          },
          {
            value: "морских миль"
          },
          {
            value: "кабельтов"
          },
          {
            value: "фурлонгов"
          },
          {
            value: "чейнов"
          },
          {
            value: "родов"
          },
          {
            value: "ярдов"
          },
          {
            value: "футов"
          },
          {
            value: "хэндов"
          },
          {
            value: "барликорнов"
          },
          {
            value: "дюймов"
```

```
      },
      {
        value: "линий"
      },
      {
        value: "точек"
      },
      {
        value: "мил"
      }
    ],
    backgroundColor: "gray",
    hoverBackgroundColor: "gray",
    border: "black"
  },
  configRightMetr: {
    placeholder: "метров",
    options: [
      {
        value: "метров"
      },
      {
        value: "сантиметров"
      },
      {
        value: "километров"
      },
      {
        value: "миллиметров"
      }
    ],
    backgroundColor: "gray",
    hoverBackgroundColor: "gray",
    border: "black"
  },
  configLeftMetr: {
    placeholder: "метров",
    options: [
      {
        value: "метров"
      },
      {
        value: "сантиметров"
      },
      {
        value: "километров"
      },
      {
        value: "миллиметров"
      }
    ],
    backgroundColor: "gray",
    hoverBackgroundColor: "gray",
    border: "black"
```

```
        }
      }
    },
    components: {
      VueDropdown
    }
    ,
    methods: {
      configLeftEngFunction(event) {
        this.$data.configLeftEng.placeholder = event.value
      },
      configRightMetrFunction(event) {
        this.$data.configRightMetr.placeholder = event.value
        this.$data.outputMetr = "0,0"
      },
      configLeftMetrFunction(event) {
        this.$data.configLeftMetr.placeholder = event.value
      },
      configRightEngFunction(event) {
        this.$data.configRightEng.placeholder = event.value
        this.$data.outputEng = "0,0"
      },
      convert() {
        axios.get('/convert/to/meter', {
          params: {
            inputValue: this.$data.inputEng,
            inputMeasurement: this.$data.configLeftEng.placeholder,
            outputMeasurement: this.$data.configRightMetr.placeholder,
          }
        }).then(response => {
          this.$data.outputMetr = response.data.result;
        }).catch(error => {
          console.log('ERROR: ' + error.response.data);
        })
        axios.get('/convert/to/eng', {
          params: {
            inputValue: this.$data.inputMetr,
            inputMeasurement: this.$data.configLeftMetr.placeholder,
            outputMeasurement: this.$data.configRightEng.placeholder,
          }
        }).then(response => {
          this.$data.outputEng = response.data.result;
        }).catch(error => {
          console.log('ERROR: ' + error.response.data);
        })
      }
    }
  }
</script>
```

CSS стили

```
<style>
  .converter {
    margin-left: 25%;
    margin-top: 5%;
  }

  .oneRaw {
    display: flex;
  }

  .pEqual {
    font-size: 25px;
    margin: 5px 10px;
    width: 30px;
  }

  .dropdownStyle {
    width: 200px;
    margin: 5px 10px;
  }

  .inputStyle {
    width: 150px;
    margin-bottom: 5px;
    margin-top: 5px;
  }

  .headerStyle {
    margin-bottom: 50px;
    margin-left: 25%;
  }

  .buttonStyle {
    margin-top: 50px;
    margin-left: 25%;
  }
</style>
```

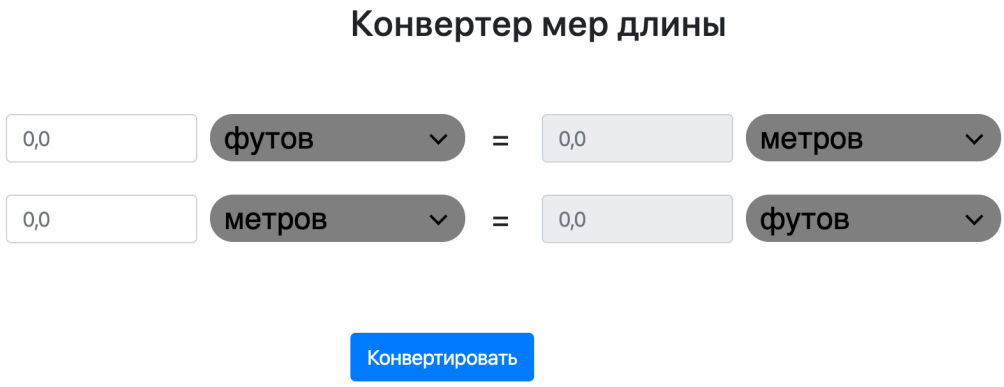
4. Пример работы программы

Скомпилированный код Web-сервера вместе с встроенным Tomcat входит в web-app.jar. Туда же входит скомпилированный код клиента. Поэтому для запуска Web-приложения необходимо просто запустить приложение командой:

```
java -jar web-app.jar
```

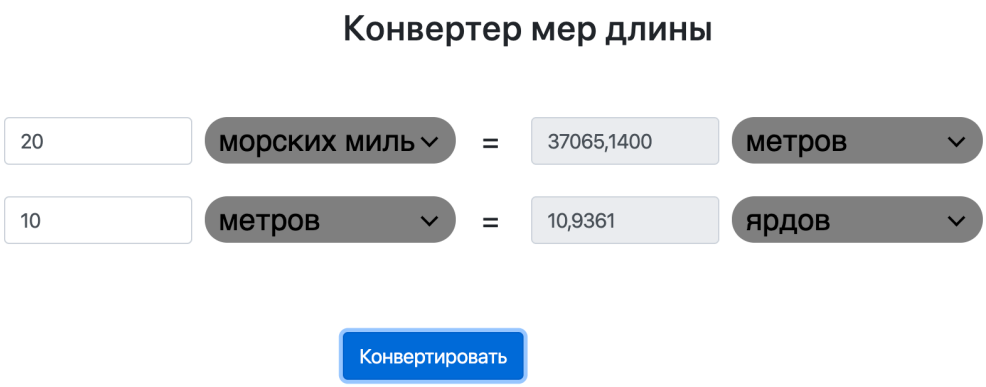
После выполнения данной команды запустится Web-сервер и будет доступен по адресу http://localhost:8080/.

После переходы по ссылке будет доступен Web-интерфейс приложения:



В поля ввода слева вводятся значения и из выпадающего списка слева и справа выбираются меры длины, после чего необходимо нажать кнопку “Конвертировать”.

Ниже приведен пример одного запроса:



5. Исходный код

Класс ConverterController

```
package com.github.sindicat.lab5.controller

import com.github.sindicat.lab5.measures.EngMeasure
import com.github.sindicat.lab5.measures.MeterMeasure
import com.github.sindicat.lab5.measures.MeterMeasurementAware
import com.github.sindicat.lab5.vo.ConversionResult
import org.springframework.web.bind.annotation.GetMapping
import org.springframework.web.bind.annotation.RequestParam
import org.springframework.web.bind.annotation.RestController
import java.math.BigDecimal
import java.math.RoundingMode

@RestController
class ConverterController {

  @GetMapping("/convert/to/eng")
  fun convertToEng(
      @RequestParam(value = "inputValue") inputValue: String,
      @RequestParam(value = "inputMeasurement") inputMeasurement: String,
      @RequestParam(value = "outputMeasurement") outputMeasurement: String
  ): ConversionResult {
    val preProcessedInputValue = inputValue.replace(',','.')
    val inputMeasure: MeterMeasurementAware =
MeterMeasure.getMeterValueByName(inputMeasurement)
    val outputMeasure: MeterMeasurementAware =
EngMeasure.getMeterValueByName(outputMeasurement)

    val outputValue = countResult(BigDecimal(preProcessedInputValue), inputMeasure, outputMeasure)
    return ConversionResult(outputValue.toString())
  }

  @GetMapping("/convert/to/meter")
  fun convertToMeter(
      @RequestParam(value = "inputValue") inputValue: String,
      @RequestParam(value = "inputMeasurement") inputMeasurement: String,
      @RequestParam(value = "outputMeasurement") outputMeasurement: String
  ): ConversionResult {
    val preProcessedInputValue = inputValue.replace(',','.')
    val inputMeasure: MeterMeasurementAware = EngMeasure.getMeterValueByName(inputMeasurement)
    val outputMeasure: MeterMeasurementAware =
MeterMeasure.getMeterValueByName(outputMeasurement)

    val outputValue = countResult(BigDecimal(preProcessedInputValue), inputMeasure, outputMeasure)
    return ConversionResult(outputValue.toString())
  }

  private fun countResult(value: BigDecimal, inputMeasure: MeterMeasurementAware, outputMeasure:
MeterMeasurementAware): BigDecimal {
    val inputValueInMeters = value * inputMeasure.meterValue
    val outputValue = inputValueInMeters.divide(outputMeasure.meterValue, 4, RoundingMode.HALF_UP)
    return outputValue.apply {
      setScale(4)
    }
  }
}
```

Класс EngMeasure

```
package com.github.sindicat.lab5.measures

import java.math.BigDecimal

enum class EngMeasure(val value: String, override val meterValue: BigDecimal) :
MeterMeasurementAware {

   LEAGUE("лиг", BigDecimal(4828.032)),
   NAUTICAL_MILE("морских миль", BigDecimal(1853.257)),
   CABLE("кабельтов", BigDecimal(185.3182)),
   FURLONG("фурлонгов", BigDecimal(201.168)),
   CHAIN("чейнов", BigDecimal(20.1168)),
   ROD("родов", BigDecimal(5.0292)),
   YARD("ярдов", BigDecimal(0.9144)),
   FOOT("футов", BigDecimal(0.3048)),
   HAND("хэндов", BigDecimal(0.1016)),
   BARLEYCORN("барликорнов", BigDecimal(0.0084667)),
   INCH("дюймов", BigDecimal(0.0254)),
   LINE("линий", BigDecimal(0.0021167)),
   POINT("точек", BigDecimal(0.000353)),
   MIL("мил", BigDecimal(0.0000254));

   companion object {
     private val meterValueByName:Map<String, EngMeasure> = values()
           .map { it.value to it }
           .toMap()

      fun getMeterValueByName(name: String): EngMeasure = meterValueByName[name] ?:
error("Unknown enum name $name")
   }

}
```

Класс MeterMeasure

```
package com.github.sindicat.lab5.measures

import java.math.BigDecimal

enum class MeterMeasure(val value: String, override val meterValue: BigDecimal):
MeterMeasurementAware {

   METER("метров", BigDecimal.ONE),
   CENTIMETER("сантиметров", BigDecimal(0.01)),
   KILOMETER("километров", BigDecimal(1000.0)),
   MILLIMETER("миллиметров", BigDecimal(0.001));

   companion object {
     private val METER_VALUE_BY_NAME:Map<String, MeterMeasure> = values()
           .map { it.value to it }
           .toMap()

      fun getMeterValueByName(name: String): MeterMeasure = METER_VALUE_BY_NAME[name] ?:
error("Unknown enum name $name")
   }
}
```

Вывод

В процессе выполнения лабораторной работы были изучены основы разработки клиент-серверных Web-приложений с использованием технологий Spring Boot и Vue JS, а также разработан Web-сервер и Web-клиент для него.