МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

Кафедра компьютерных технологий и программной инженерии

ОТЧЁТ
ЗАЩИЩЁН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ассистент		Кочин Д. А.
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Отчёт по лабораторной работе №2

по курсу: Технология разработки серверных информационных систем

СТУДЕНТ ГР. № 4932 (номер группы) _____ (подпись, дата) С. И. Коваленко (инициалы, фамилия)

Санкт-Петербург
2021

1. Цель работы:

Разработка ресурса REST/JSON сервиса

2. Вариант:

9. Складской учёт.

3. Описание:

Функциональные требования к системе складского учёта:

- Возможность добавлять новые помещения
- Возможность удалять помещения
- Возможность просмотра всех имеющихся помещений
- Возможность добавлять объекты в помещения
- Возможность удалять объекты из помещений
- Возможность просмотра всех предметов в помещении

4. Вывод:

В ходе выполнение лабораторной работы было создано приложение на языке kotlin с использованием технологии сервлетов. В качестве сервера использовано tomcat 9.0.53.

Приложение А

```
// @filename /src/main/kotlin/com/example/demo/DemoApplication.kt
package com.example.demo

import org.springframework.boot.autoconfigure.SpringBootApplication
import org.springframework.boot.runApplication

@SpringBootApplication
class DemoApplication

fun main(args: Array<String>) {
   runApplication<DemoApplication>(*args)
}


// @filename /src/main/kotlin/com/example/demo/ServletInitializer.kt
package com.example.demo

import org.springframework.boot.builder.SpringApplicationBuilder
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer

class ServletInitializer : SpringBootServletInitializer() {

   override fun configure(application: SpringApplicationBuilder): SpringApplicationBuilder {
      return application.sources(DemoApplication::class.java)
   }

}


// @filename /src/main/kotlin/com/example/demo/controller/ObjController.kt
package com.example.demo.controller

import com.example.demo.data.Storage
import com.example.demo.data.objFactory
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.*

@RestController
@RequestMapping("/obj")
class ObjController {
   /** добавление объекта в комнату */
   @PostMapping("/add")
   fun addObj(
      @RequestParam("name") name: String,
      @RequestParam("count") count: Int,
      @RequestParam("roomId") roomId: Int) {
      val obj = objFactory(name, count)
```

```
    if ( obj != null ) {
        Storage.data.roomList
            .filter { it.id == roomId }
            .forEach { it.put(obj) }

        ResponseEntity
            .ok()
            .body("successfully")
    }
    else
        ResponseEntity
            .badRequest()
            .body("incorrect data")
}

/** Изменение количества объектов */
@PutMapping("/put")
fun putObj(
    @RequestParam("objName") objName: String,
    @RequestParam("roomId") roomId: Int,
    @RequestParam("count") count: Int): ResponseEntity<String> {
    var f = false
    for ( room in Storage.data.roomList)
        if ( room.id == roomId )
            for ( obj in room.listObj )
                if ( obj.name == objName ) {
                    f = true
                    obj.count = count
                }
    return if ( f )
        ResponseEntity
            .ok()
            .body("successfully")
    else
        ResponseEntity
            .badRequest()
            .body("error")
}

/** Удаление объекта */
@DeleteMapping("/delete")
fun deleteObj(
    @RequestParam("objName") objName: String,
    @RequestParam("roomId") roomId: Int): ResponseEntity<String> {
    var f = false
    for ( room in Storage.data.roomList)
        if ( room.id == roomId )
            for ( i in 0 until room.listObj.size )
```

```
                if ( room.listObj[i].name == objName ) {
                    f = true
                    room.listObj.removeAt(i)
                }
        return if ( f )
            ResponseEntity
                .ok()
                .body("successfully")
            else
            ResponseEntity
                .badRequest()
                .body("error")
    }
}

// @filename /src/main/kotlin/com/example/demo/controller/RoomController.kt
package com.example.demo.controller

import com.example.demo.data.Storage
import com.example.demo.data.erroreStorage
import com.example.demo.data.objFactory
import com.google.gson.GsonBuilder
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.*

@RestController
@RequestMapping("/room")
class RoomController {

    val gsonBuilder = GsonBuilder() .setPrettyPrinting() .create()

    /** Получение списка комнат */
    @GetMapping("/get")
    fun getRoomList() =
        ResponseEntity
            .ok()
            .body( gsonBuilder.toJson(Storage.data) )

    /** Создание новой комнаты */
    @PostMapping("/add")
    fun addRoom(@RequestParam("name") name: String,@RequestParam("id") id: Int) =
        if ( Storage.addRoom(name, id) )
            ResponseEntity
            .ok()
            .body("successfully")
        else
            ResponseEntity
            .badRequest()
```

```
            .body(erroreStorage.lastError)


    /** Удаление комнаты */
    @DeleteMapping("/delete")
    fun deleteRoom(@RequestParam("id") id: Int) =
        if  ( Storage.deleteRoom(id) )
            ResponseEntity
            .ok()
            .body("successfully")
        else
            ResponseEntity
            .badRequest()
            .body(erroreStorage.lastError)

}

// @filename /src/main/kotlin/com/example/demo/data/erroreStorage.kt
package com.example.demo.data

object erroreStorage {
    var lastError = ""

    /** сообщить о возникновении ошибки **/
    fun error(error: String) {
        println(error)
    }

    /** возвращает текст последней ошибки **/
    fun getError() = lastError
}

// @filename /src/main/kotlin/com/example/demo/data/Obj.kt
package com.example.demo.data

data class Obj(val name: String, var count: Int)

// @filename /src/main/kotlin/com/example/demo/data/objFactory.kt
package com.example.demo.data

fun objFactory(name: String, count: Int) =
    if ( name.isNotEmpty() && count >= 0 )
        Obj(name, count)
    else
        null

// @filename /src/main/kotlin/com/example/demo/data/Room.kt
package com.example.demo.data
```

```
data class Room(val name: String, val id: Int) {
    val listObj = mutableListOf<Obj>()

    /** колличество наименований объектов **/
    fun size() = listObj.size

    /** добавить объекты в комнату **/
    fun put(obj: Obj) {
        for (o in listObj)
            if (o.name == obj.name) {
                o.count + obj.count
                return
            }
        listObj.add(obj)
    }

    /** выгрузить объекты из комнаты **/
    fun get(name: String, count: Int) {
        for (o in listObj) {
            if ( o.name == name )
                if ( o.count < count )
                    println("Error insufficient quantity in stock")
                else
                    o.count - count
        }
    }

}

// @filename /src/main/kotlin/com/example/demo/data/Storage.kt
package com.example.demo.data

object Storage {
    val data = storageData()

    /** колличество комнат **/
    fun size() = data.roomList.size

    /** добавление комнаты на склад
     * true - успешное добавлени**/
    fun addRoom(name: String, id: Int) : Boolean {
        if (name.isNotEmpty() && id >= 0) {
            for (r in data.roomList)
                if (r.name == name || r.id == id) {
                    erroreStorage.error("Error room with such data already exists")
                    return false
                }
```

```
            data.roomList.add(Room(name, id))
            return true
        } else {
            erroreStorage.error("Error directional data about the room")
            return false
        }
    }


    /** удаление комнаты **/
    fun deleteRoom(id: Int): Boolean {
        for (i in 0 until data.roomList.size) {
            if ( data.roomList[i].id == id ) {
                data.roomList.removeAt(i)
                return true
            }
        }
        erroreStorage.error("Error this room not found")
        return false
    }

}

// @filename /src/main/kotlin/com/example/demo/data/storageData.kt
package com.example.demo.data

open class storageData {
    val roomList = mutableListOf<Room>()
}

// @filename /src/test/kotlin/com/example/demo/DemoApplicationTests.kt
package com.example.demo

import org.junit.jupiter.api.Test
import org.springframework.boot.test.context.SpringBootTest

@SpringBootTest
class DemoApplicationTests {

    @Test
    fun contextLoads() {
    }

}
```