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

КУРСОВОЙ ПРОЕКТ
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

старший преподаватель		Е. О. Шумова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ

Расписание занятий

по дисциплине: ОБЪЕКТНО ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4932		С. И. Коваленко
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2021

Содержание

Задание на курсовое проектирование

Разработка системы классов «Расписание занятий в университете» (В системе должны поддерживаться режимы поиска занятия по заданному критерию (номер группы, преподаватель), регистрации занятий)

Введение

Предметная область «расписание занятий» очень большая и может быть рассмотренная с разных сторон и для разных ролей. Расписание разных организаций имеет не только разное содержание, но и структуру. Из-за большой вариативности реализаий я остановился на расписании нашего вуза.

Расписание нашего вуза имеет сложную структуру. Занятия проходят не только с понедельника по пятницу, но есть часть занятий вне сетки расписания (без фиксированного времени). В одно время, но на разных неделях(по чётности) могут быть разные занятия. Одному занятию может быть назначено несколько преподавателей, аудиторий и групп.

Дополнительной сложностью в данной предметной области является быстрое изменение данных. Исправления в расписание вносятся на протяжении всего семестра. Но пользователю, будь то студент или преподаватель очень важно получать актуальную информацию. По этому авто обновление является необходимой опцией для данной предметной области.

1. Постановка задачи

1.1. Анализ предметной области

Как уже сказано выше данная предмета область очень обширна и даже её сужение до расписания конкретного вуза оставляет множество объектов.

Расписанием будем считать список дней и критерий их объединяющий, например номер группы и список дней с занятиями. Для описания одного дня нам тоже потребуется сущность хранящая список пар в этот день и названия дня.

Под парой будем понимать одно или два занятия расположенные в одно время. Тогда «пара» должна хранить время начала и конца занятия, как дополнительную информацию можно хранить номер занятия.

Занятие, в свою очередь, должно хранить: название и тип занятия(лр, л, кп,

пр), а так же списки: преподавателей ведущих занятие, групп и аудиторий.

Из анализа предметной области следует что должны быть сущности: расписание группы, расписание преподавателя, день, пара, занятие и преподаватель.

1.2. Формулировка технического задания

Реализовать приложение расписания вуза. Приложение должно иметь графический интерфейс. Загрузка и обновление данных должны выполняться автоматически во всех ситуациях когда данные по той или иной причине перестали быть актуальными. Для отображения расписания пользователь должен выбрать поле поиска, ввести туда либо
номер группы, либо номер аудитории, либо ФИО преподавателя. В реультате этих действий система должна вывести расписание удовлетворяющее входным данным. Для повышения скорости загрузки приложения или данных по сети необходимо реализовать локальное хранение данных прошлой сессии использования.

2. Проектирование классов

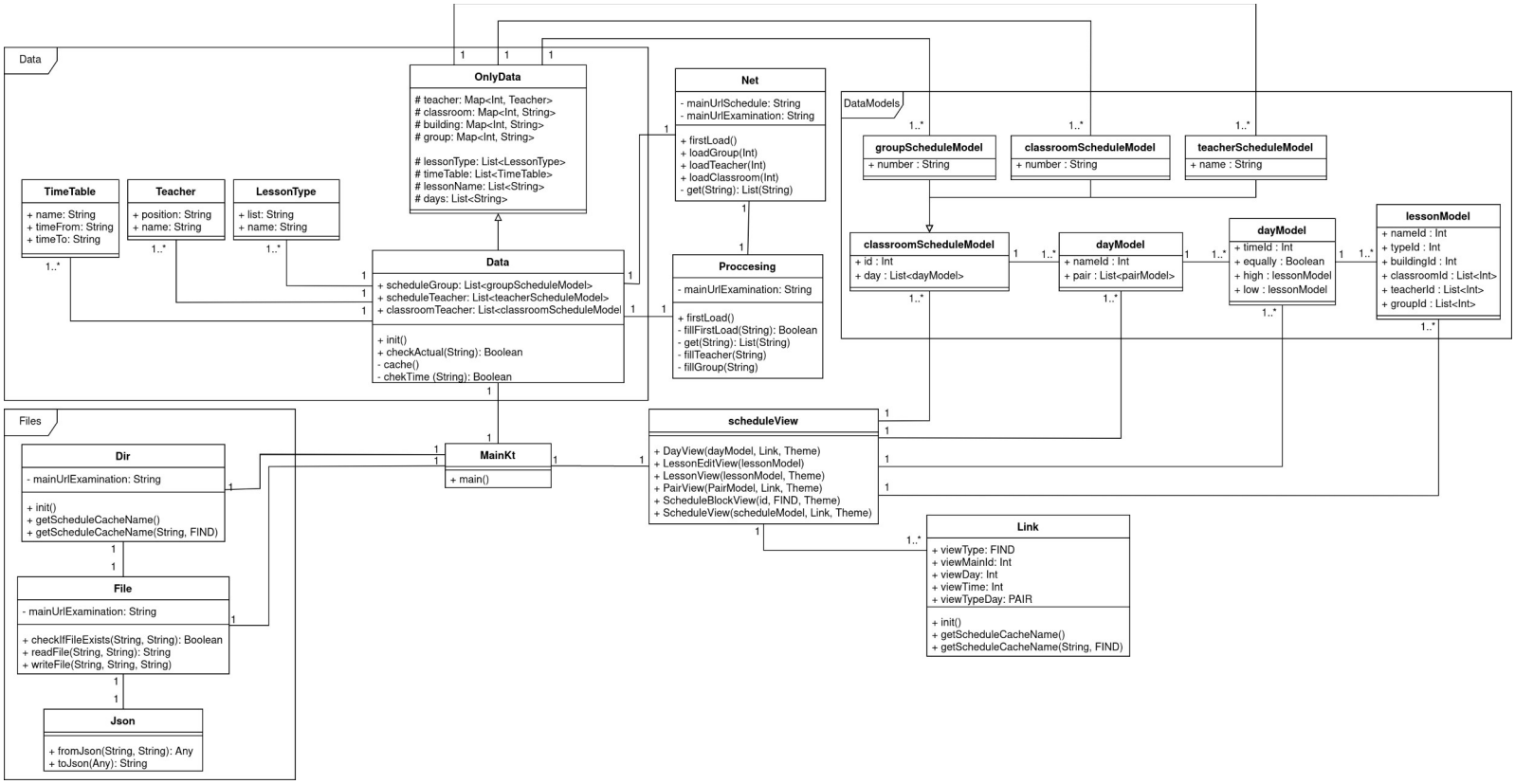

Рис. 1 UML-диаграмма классов

2.1. Классы сущностей

Название	Данные хранимые в данной сущности
OnlyData	Список - объектов Teacher(ФИО, должность) Список - номеров аудиторий Список - адресов зданий где проводятся занятия Список - номеров групп
Schedule	Id данного расписания Список объектов - Day(хранящих расписание дня)
ScheduleClassroom	Сущность наследуемая от Schedule Строка - номер аудитории
ScheduleGroup	Сущность наследуемая от Schedule Строка - номер группы
ScheduleTeacher	Сущность наследуемая от Schedule Строка - ФИО преподавателя
Day	Число - номер дня Список объектов — Pair(хранящих расписание пары)
Pair	Сущность хранящая данные о паре в расписании
lessonModel	Сущность хранящая данные об одном занятии
Link	Сущность хранящая хранящая путь до элемента в расписании

2.2. Управляющие классы

Название	Методы
Data	CheckActual - проверяет актуальность расписания addSchedule - добавлет расписание в хранилище init - загружает данные из прошлой сессии и проверяет их актуальность cache — сохраняет данные текущей сессии
Net	LoadTeacher — загружает расписание преподавателя loadGroup - загружает расписание группы loadClassroom - загружает расписание аудетории
findManager	ProcessingString — определяет какой тип поиска используее пользователь findGroup — подготавливает данные и начинает загружать расписание группы findTeacher - подготавливает данные и начинает загружать

	расписание преподавателя findClassroom -подготавливает данные и начинает загружать расписание аудетории
htmlManager	FillFirstLoad — получае первичные данные из запроса fillGroup — заполняет расписание группы из данных полученных с сервера

2.3. Интерфейсные классы

Название	Описание
ScheduleBlockView	Отвечает за отображение всего блока расписания и выбор его типа
DayView	Отвечает за отображение одного дня
LessonView	Отвечает за отображение времени(одна или две пары)
PairView	Отвечает за отображение одного занятия
ScheduleView	Отвечает за отображение всего блока расписания
FindView	Отвечает за отображение поисковой строки
TopBarView	Отвечает за отображение верхней части экрана

2.4. Используемые паттерны проектирования

Паттерн	Классы его реализующие
Одиночка (`Singleton`)	Data, findManager, Net, htmlManager
Прототип (Clon)	Link
Состояние (state)	LESSON_TYPE, LT_Lecture, LT_Practical, LT_Laboratory, LT_CourseProject, LT_CourseWork

Одиночка

Все реализации одиночки сводятся к тому, чтобы скрыть конструктор по умолчанию и создать публичный статический метод, который и будет

контролировать жизненный цикл объекта-одиночки.

Если у вас есть доступ к классу одиночки, значит, будет доступ и к этому статическому методу. Из какой точки кода вы бы его ни вызвали, он всегда будет отдавать один и тот же объект.

В kotlin есть встроенная реализация данного шаблона, для этого необходимо вместо колючеголовые class указать слово object.

```
object Data : OnlyData() {

object Net {

object htmlManager {

object findManager {
```

Прототип

Паттерн Прототип поручает создание копий самим копируемым объектам. Он вводит общий интерфейс для всех объектов, поддерживающих клонирование. Это позволяет копировать объекты, не привязываясь к их конкретным классам.

Данный шаблон используется в классе Link, который реализует адресацию до элемента.

```
data class Link(
    var viewType: FIND = FIND.NOPE,
    var viewMainId: Int = 0,
    var viewDay: Int = 0,
    var viewTime: Int = 0,
    var viewTypeDay: PAIR_TIME = PAIR_TIME.EQUALLY
){
    /** Возвращает, копию ссылки **/
    fun getCopy() = Link(viewType,viewMainId,viewDay,viewTime,viewTypeDay)
}
```

Состояние

Паттерн Состояние предлагает создать отдельные классы для каждого состояния, в котором может пребывать объект, а затем вынести туда поведения, соответствующие этим состояниям.

```
data class LT_Lecture(override val liter: String = "Л", override val name: String = "лекция") : LESSON_TYPE("","лекция")
data class LT_Practical(override val liter: String = "ПР", override val name: String = "практическое занятие или семинар") : LESSON_TYPE("", "")
data class LT_Laboratory(override val liter: String = "ЛР", override val name: String = "лабораторные занятия") : LESSON_TYPE("", "")
```

```
data class LT_CourseProject(override val liter: String = "КП", override val name: String = "курсовой проект") : LESSON_TYPE("", "")
data class LT_CourseWork(override val liter: String = "КР", override val name: String = "курсовая работа") : LESSON_TYPE("", "")
```

3. Разработка приложения

3.1. Разработка интерфейса приложения

Интерфейс программы логически должен быть разделён на 2-е части. Первая отвечает за поиск и вторая за отображение информации.

В первой части можно отказаться от кнопок и оставить только одну поисковую строку. При анализе ввода можно 100% понять, что именно хочет найти пользователь.

```
fun DetermineSearchType(str: String) : FIND {
   return when {
      "-".toRegex().containsMatchIn(str) -> FIND.CLASSROOM
      "(\\d){4}".toRegex().containsMatchIn(str) -> FIND.GROUP
      !"""[${'$'}M${'$'}K${'$'}KB${'$'}KC${'$'}ВЦ${'$'}MK$
{'$'}КСВ^А^В^И^Md]""".toRegex().containsMatchIn(str) -> FIND.TEACHER
      else -> FIND.NOPE
   }
}
```

Данная функция проанализировав введённую строку, вернёт тип поиска.

Во второй части будет выводиться расписание. Расписание представляет список дней. Каждый день содержит список «времён» и название дня. «Время» имеет время начала и конца занятия и несколько «пар»(на верхней и нижней неделях). «Пара» содержит информацию: тип занятия, название, название дисциплины, список ФИО преподавателей, список номеров групп, список номеров аудиторий и адрес проведения занятий. По этому будет логичным расположить список дней вертикально. Находиться пара на нижней или верхней неделе, лучше отображать цветом. Самой важной информацией является название, время и место занятия, поэтому их можно сделать более контрастными.

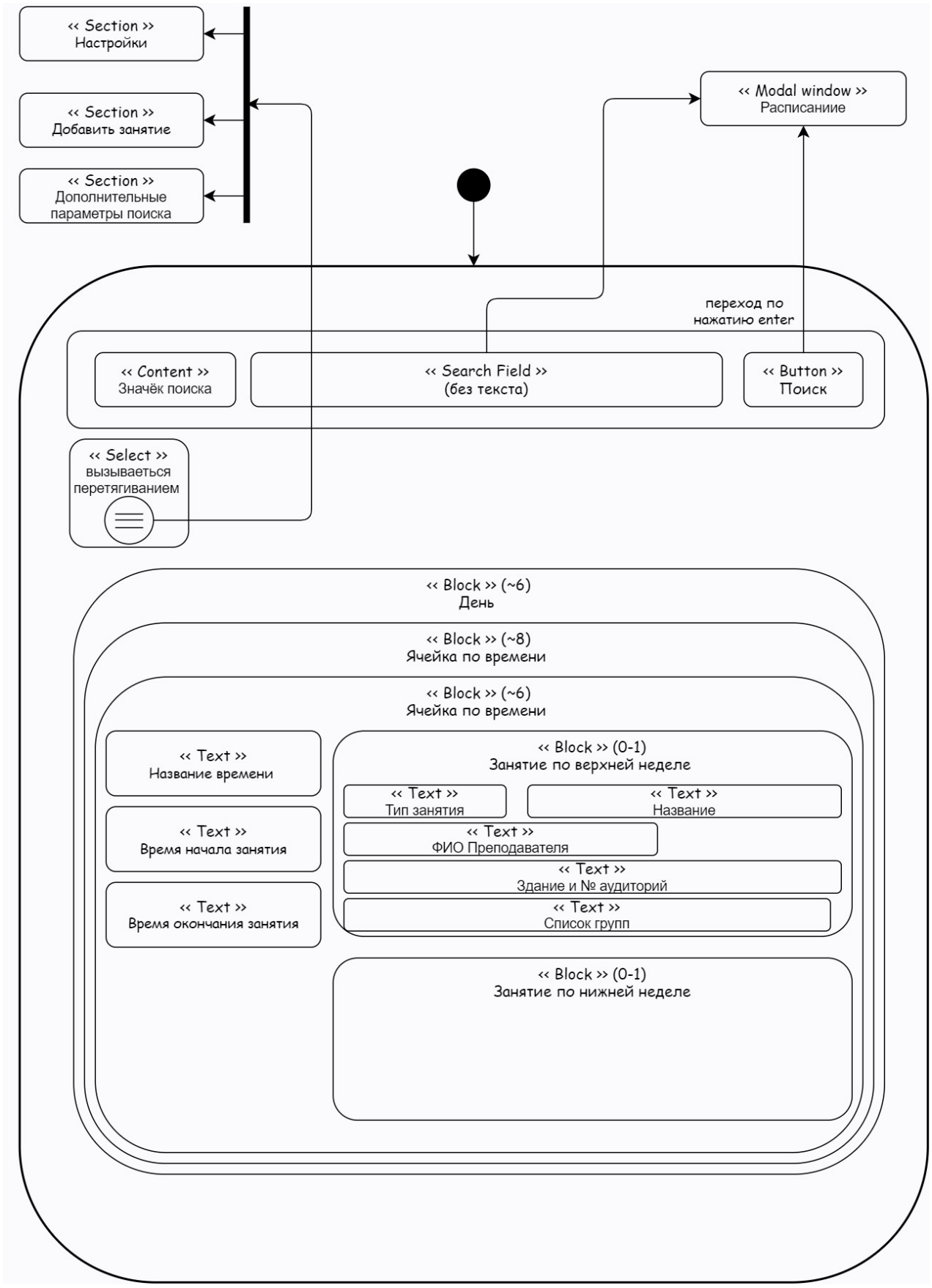

Рис. 2 UML-диаграмма структуры главного экрана

3.2. Реализация классов

Основные логические функции

Название	Параметры	Возвращаемое значение	Описание
findGroup	number: String	---	Загружает данные расписания с использованием кэша если там есть данная группа, иначе выполняется запрос на сервер.
processingString	str: String	---	Обрабатывает строку и вызывает загрузку расписания.
fillTeacher	html: String	Boolean	Создаёт и заполняет объект расписания преподавателя, из html кода страницы true – корректное создание.
fillFirstLoad	html: String	Boolean	Заполняет списки преподавателей, групп и аудиторий. Выполняется каждый раз при изменении расписания.
scheduleFromText	html: String	---	Заполняет расписание из html кода страницы.

Процесс поиска группы

Пользователь вводит запрос, его обрабатывает processingString и вызывает findGroup. Если данные были а кэше произойдёт их загрузка.

```
if ( checkIfFileExists(Dir.cache, fileName) ) {     // Проверка, есть ли данная группа в кэше
    val fullGroup = loadScheduleFromCache(fileName, ScheduleGroup())   // Загруска из кэша
```

Иначе вызывается loadGroup выполняющая загрузку

Процесс поиска преподавателя

Основная часть похожа, но processingString вызывает getListSimilarTeacher и получает список преподавателей подходящих под запрос пользователя.
Если в списке всего 1 преподаватель, то выполняется поиск его расписания findTeacher.

Процесс вывода расписания

ScheduleBlockView определяет какое расписание сейчас нужно отобразить. Получает это расписание из хранилища

```
val teacher = Data.getTeacher(id.value)
```

и передаёт его функции *ScheduleView вызывает вывод с помощью функции DayView для каждого дня*

```
for (i in 0 until schedule.day.size) {
    Box(
        Modifier.background(theme.value.day)
    ) {
        DayView(schedule.day[i].toFull(), schedule.day[i], link.getCopy(), theme)
    }
```

Процесс обработки ответа от сервера

Анализ полученных данных, проверка на корректность и запуск заполнения.

```
fun fillTeacher(html: String) =
    if ( Data.checkActual(html) ) {
        val number = html.substringAfter("Расписание для преподавателя - ")
            .substringBefore("</h2>")
        val htmlSchedule = html.substringAfter("""</h2>""")
            .substringBefore("</div></div></div></div>")
        logger.info { "fillTeacher \n$html" }
        // преобразование html представления РАСПИСАНИЯ ГРУППЫ в объект и сохраняет его
        scheduleFromText(htmlSchedule, number, ScheduleTeacher())
        true
    } else {
        logger.error { "loadGroup request error" }
        false
    }
```

Заполнение расписания с вызовом dayFromText

```
/** общая функция для заполнения расписаний **/
private fun scheduleFromText(html: String, name: String, schedule: Schedule) {
    when (schedule) {
        is ScheduleGroup -> {
            schedule.number = name
            schedule.id     = Data.getGroupId(name)
        }
        is ScheduleTeacher -> {
            schedule.name = name
            schedule.id = getTeacherId(name.substringBefore(" - "))
```

```
        }
        is ScheduleClassroom -> {
            // not testing
            schedule.number = name
            schedule.id = getClassroomId(name)
        }
    }
    if ( schedule.id == -1 )
        logger.error { "scheduleFromText schedule id = -1 for [$name]" }
        html.split("<h3>").forEach {
            val day = dayFromText(it)
            day?.let { Day -> schedule.day.add(Day) }
        }
    if (Data.addSchedule(schedule))
        logger.error { "error add new schedule to data" }
    logger.info { schedule.toString() }
}
```

Преобразование дня в объект и возврат его

```
/** преобразование html представления ДНЯ в объект **/
private fun dayFromText(html: String) : Day? {
    if ( html.isEmpty() ) {
        logger.warn { "dayFromText text is empty " }
        return null
    }
    logger.info { "dayFromText $this" }
    val name = html.substringAfter("""<h3>""")
                .substringBefore("""</h3>""")
    val nameId = Data.getDayId(name)
    if ( nameId == -1 ) {
        logger.error { "dayFromText Non-existent day[$name]" }
        return null
    }
    val day = Day(nameId)
    html.substringAfter("<h4>")
        .split("<h4>")
        .forEach {
        day.pair.add(pairFromText(it))
    }
    return day
}
```

Преобразования пары

```
/** преобразование html представления ПАРЫ в объект **/
private fun pairFromText(html: String) : Pair {
    logger.info { "lessonFromText [$html]" }
    val pair = Pair()
```

```
    pair.timeId = Data.getTimeId(
        html.substringAfter("<h4>")
            .substringBefore("""</h4>""")  // Дата
    )
    val flagUp = """(class="up")""".toRegex().containsMatchIn(html)
    val flagDn = """(class="dn")""".toRegex().containsMatchIn(html)
    if ( flagUp && flagDn ) {                    // Мигалка
        pair.equally = false
        pair.high =
            fillLessonFromString(
                html.substringBefore("""<div class="study"><span><b class="dn"""")
            )
        pair.low  = fillLessonFromString(
            html.substringAfter("""</a></span></div>""")
        )
        logger.info { "lessonFromText two [${pair.high}] [${pair.low}]" }
    } else if ( !flagUp && !flagDn ) {           // Одинаковое
        pair.equally = true
        pair.high = fillLessonFromString(html)
        logger.info { "lessonFromText one [${pair.high}]" }
    } else if ( flagUp ) {                       // Только верхняя
        pair.equally = false
        pair.high = fillLessonFromString(html)
        logger.info { "lessonFromText up [${pair.high}]" }
    } else {                                     // только нижняя
        pair.equally = false
        pair.low = fillLessonFromString(html)
        logger.info { "lessonFromText down [${pair.low}]" }
    }
    return pair
}
```

С помощью рекурсивного заполнения достигается простота и надёжность заполнения.

Хранение объектов

Для удобства и скорости программы у всех сущностный связанных с расписанием есть две версии. Первая содержит id всех внутренних объектов. Вторая непосредственно объекты.

```
/** Один день, одной группы **/
data class Day(
    /** ID дня **/
    val nameId: Int = 0
) {
    /** Список занятий на день **/
    var pair = mutableListOf<Pair>()
```

```
    fun toFull() : fullDayModel{
        val fullDay = fullDayModel(Data.getDayName(nameId))
        pair.forEach { fullDay.pair.add( it.toFull() ) }
        return fullDay
    }
}
```

и

```
data class fullDay(
    /** Название дня **/
    val name: String = ""
) {
    /** Список занятий на день **/
    var pair = mutableListOf<fullPair>()
}
```

в частях программы не связанных с выводом и смыслом содержимого используются версии с id, они меньше весят и быстрее обрабатываются.

4. Тестирование

Поиск расписания группы

1) Запустив приложение пользователь видит экран

рис. 3 стартовый экран

2) Пользователь вводит номер группы в поисковую строку, система автоматически загружает и отображает расписание

рис. 4 расписание группы 4932

3) Пользователь может листать расписание

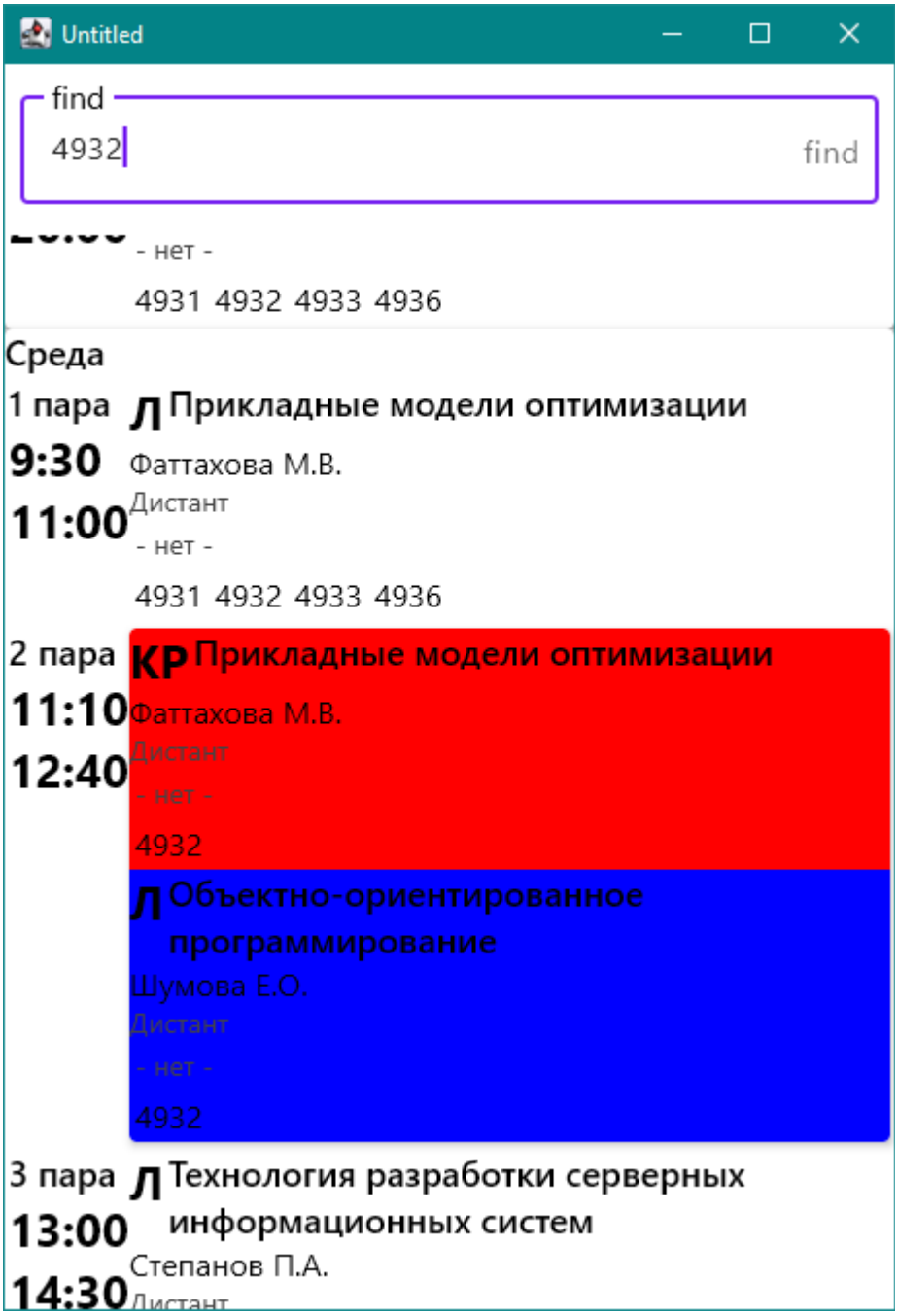

рис. 5 перемещение по расписанию

Поиск расписания преподавателя

1) Запустив приложение пользователь видит экран (рис. 3)
2) Пользователь вводит фамилию преподавателя в поисковую строку(как только введённой части достаточно для однозначной идентификации преподавателя) система автоматически загружает и отображает расписание

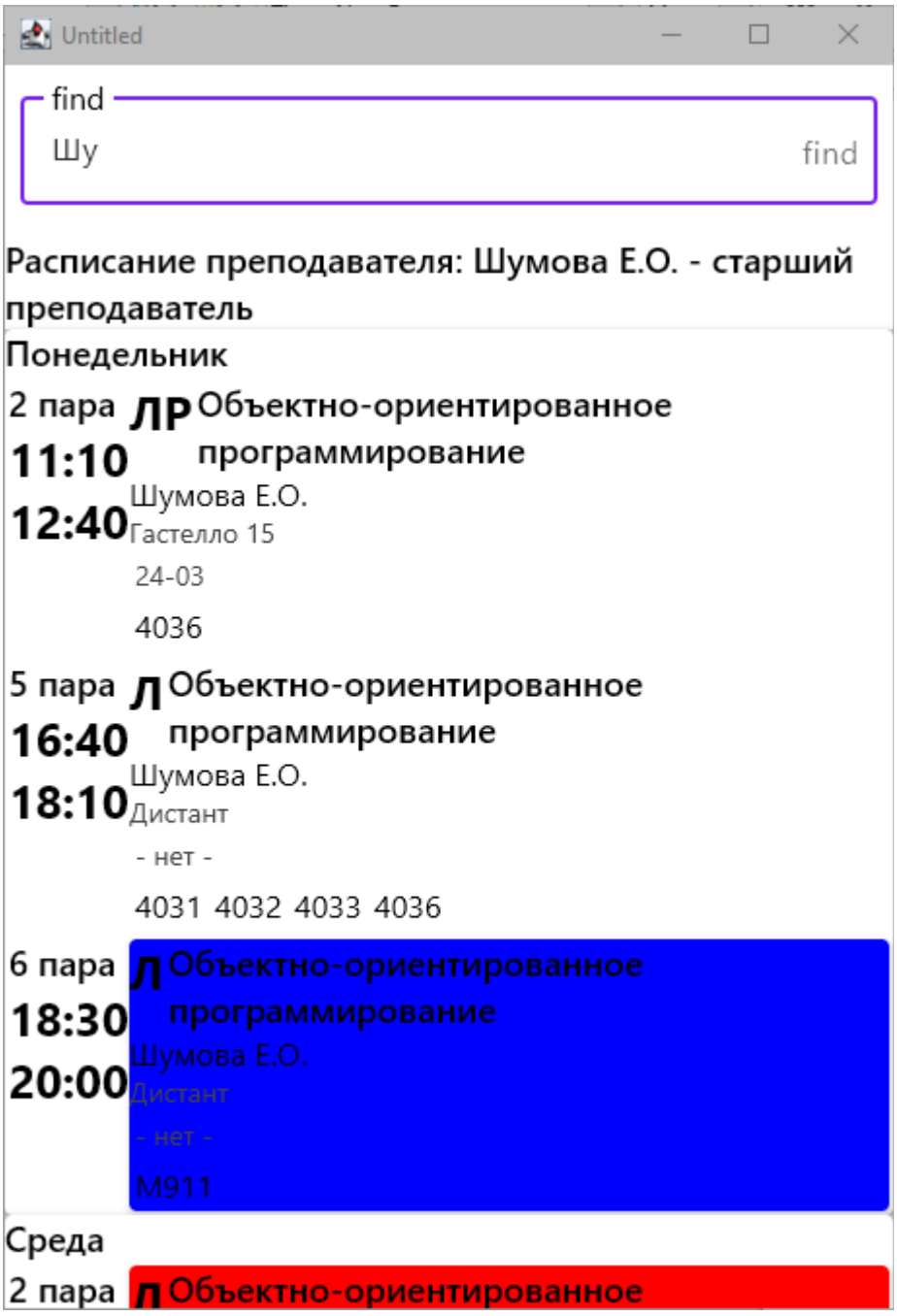

рис. 6 вывод расписания преподавателя

3) Пользователь может листать расписание

рис. 7 Перемещение по расписанию преподавателя

Заключение

В результате выполнения курсового проекта было реализовано приложение для просмотра расписания и закреплена информация полученная в курсе по объектно ориентированному программированию.

Список использованных источников

1. https://kotlinlang.org/
2. https://developer.android.com/jetpack/compose/
3. «Kotlin in Action» Жемчужеров Дмитрий, Исакова Светлана. издание, ДМК Пресс, 2018 ISBN 978-5-97060-497-7

Приложение А

```
// @filename /core/Themes.kt
package org.suai.schedule.core

import androidx.compose.ui.graphics.Color
import mu.KotlinLogging
import org.suai.schedule.core.settings.Settings
import org.suai.schedule.model.Dir
import org.suai.schedule.model.Theme
import org.suai.schedule.utils.file.readFile
import org.suai.schedule.utils.file.writeFile
import org.suai.schedule.utils.json.fromJson
import org.suai.schedule.utils.json.toJson

object Themes: ThemeStorage() {
    lateinit var actual: Theme
    val logger = KotlinLogging.logger {}

    /** Получение темы по id **/
    fun getTheme(Id: Int) : Theme? {
        logger.info { "getTheme [$Id]" }
        for ( theme in themes )
            if ( theme.id == Id )
                return theme
        return null
    }
```

```
    /** Размер списка тем **/
    fun size() = themes.size

    /** Возвращает тему по порядковому номеру **/
    fun getByNumber(number: Int): Theme {
        logger.info { "getByNumber [$number]" }
        return themes[number]
    }

    /** Устанавливает актуальную тему **/
    fun setActualTheme(newActual: Theme) {
        Settings.theme.value = newActual.id // Что за дичь с начтройками aaaaaaaAAA11!!!
        actual = newActual
    }

    /** Загруска только используемой темы, с акуратной вставкой **/
    private fun optimalLoad(): Boolean {
        logger.info { "Theme optimalLoad" }
        val loadTheme = fromJson(
            readFile(Dir.cache, "theme.json"),
            Theme( 1, "", Color.White, Color.Cyan, Color.Magenta, Color.Blue, Color.Red, Color.Yellow,
Color.LightGray, Color.Black, Color.DarkGray))
        return if (loadTheme is Theme) {
            actual = loadTheme
            merger(actual)
        } else
            false
    }

    /** кэширует актуальную тему **/
    fun cacheActualTheme() {
        writeFile(Dir.cache, "theme.json", toJson(actual))
    }

    /** Вставляет тему в список
     *  Устанавливает её как актуальную
     *  true, если темы с таким id небыло **/
    private fun merger(theme: Theme) : Boolean {
        logger.info { "merger [$theme]" }
        actual = theme
        themes.forEach { if ( it.id == actual.id ) return false }
        themes.add(actual)
```

```
    return true
}

/** Очистка хранилища,
 *  Загруска тем из файла **/
fun load(): Boolean {
    logger.info { "Theme load" }
    val loadThemes = fromJson(readFile(Dir.data, "theme.json"), ThemeStorage())
    return if ( loadThemes is ThemeStorage ) {
        this.themes = loadThemes.themes
        setActualTheme(themes[0])
        cacheActualTheme()
        true
    } else {
        false
    }
}

/** Сортирует список тем
 *  Сохраняет все темы **/
fun save() {
    logger.info { "save" }
    sort()
    val themeStorage: ThemeStorage = Themes
    writeFile(Dir.data, "theme.json", toJson(themeStorage))
}

/** Сортирует темы по id **/
private fun sort() {
    themes.sortBy { it.id }
}

/** Первичная инициализация **/
fun init() {
    logger.info { "init" }
    if ( optimalLoad() ) {
        logger.info { "init complete" }
    } else if ( load() ) {
        logger.info { "init complete" }
    } else {
        themes.add(
            Theme(
```

```
                id = 1,
                name = "Default",
                background = Color.White,
                first = Color.Cyan,
                second = Color.Magenta,
                lowLesson = Color.Blue,
                topLesson = Color.Red,
                lesson = Color.Yellow,
                day = Color.LightGray,
                font = Color.Black,
                sabFont = Color.DarkGray
            )
        )
        themes.add(
            Theme(
                id = 2,
                name = "Red",
                background = Color(0x01000000),
                first = Color(0xE8772300),
                second = Color(0xFF6B3200 ),
                lowLesson = Color(0xFFA32600),
                topLesson = Color(0xFF262C00),
                lesson = Color(0xFF6B3200),
                day = Color.LightGray,
                font = Color.Black,
                sabFont = Color.DarkGray
            )
        )
        actual = themes[0]
        cacheActualTheme()
        save()
        logger.info { "init default complete" }
    }
  }
}

// @filename /core/ThemeStorage.kt
package org.suai.schedule.core

import org.suai.schedule.model.Theme

open class ThemeStorage {
```

```
    internal var themes = mutableListOf<Theme>()
}

// @filename /core/data/Data.kt
package org.suai.schedule.core.data

import mu.KotlinLogging
import org.suai.schedule.core.net.Net
import org.suai.schedule.model.Dir
import org.suai.schedule.model.Link
import org.suai.schedule.model.schedule.*
import org.suai.schedule.utils.file.checkIfFileExists
import org.suai.schedule.utils.file.readFile
import org.suai.schedule.utils.file.writeFile
import org.suai.schedule.utils.json.fromJson
import org.suai.schedule.utils.json.toJson

object Data : OnlyData() {

    private val logger = KotlinLogging.logger {}

    @Suppress("VARIABLE_IN_SINGLETON_WITHOUT_THREAD_LOCAL")
    /** Список расписаний занятий для группы **/
    private var scheduleGroup     = mutableListOf<ScheduleGroup>()
    @Suppress("VARIABLE_IN_SINGLETON_WITHOUT_THREAD_LOCAL")
    /** Список расписаний занятий в аудитории **/
    private var scheduleClassRoom = mutableListOf<ScheduleClassroom>()
    @Suppress("VARIABLE_IN_SINGLETON_WITHOUT_THREAD_LOCAL")
    /** Список расписаний занятий для преподавателей **/
    private var scheduleTeacher   = mutableListOf<ScheduleTeacher>()

    /** проверяет актуальность расписания (true - расписание актуально)**/
    fun checkActual(htmlString: String) : Boolean {
        val flag = checkTime(htmlString.substringAfter("(Сборка: ").substringBefore(")"))
        if ( flag ) {
            // Все действия связанные с изменением расписания TODO()
            logger.error { "server schedule rebuilding  !!!" }
        }
        return flag
    }

    /** добавлет расписание в хранилище, true - успешное добавление **/
```

```
fun addSchedule(schedule: Schedule) =
    when (schedule) {
        is ScheduleGroup     -> scheduleGroup.add(schedule)
        is ScheduleTeacher   -> scheduleTeacher.add(schedule)
        is ScheduleClassroom -> scheduleClassRoom.add(schedule)
        else -> false
    }

/** Проверяет актуальность расписания (true - обновления нет)**/
private fun checkTime(newTime: String) =
    if ( time != newTime && time.isNotEmpty() ) {
        logger.warn { "checkTime schedule change [$newTime] last [${Data.time}]" }
        false
    } else
        true

/** При возможности загружает данные из кэша, иначе отправляет первичный запрос **/
fun init() {
    if ( checkIfFileExists(Dir.cache,"Data.schedule.json") ) {
        logger.info { "init" }
        val jsonString = readFile(Dir.cache,"Data.schedule.json")
        if ( jsonString.isNotEmpty() ) {
            val onlyData = fromJson(jsonString, OnlyData())
            if ( onlyData is OnlyData)
                onlyData.toData(Data)
            //else  вроде такое невозможно
            //    logger.error { "init onlyData is not OnlyData!" }
            return
        } else
            logger.error { "Load data empty file" }
    } else
        logger.warn { "File [${Dir.cache}Data.schedule.json] not exist " }
    Net.firstLoad()      // загружаем
    autoFillDays()       // дни недели
    cache()              // сохраняем
}

/** Сохранение данных в кэш **/
private fun cache() {
    logger.info { "cache" }
    val onlyData: OnlyData = Data
    writeFile(Dir.cache,"Data.schedule.json", toJson(onlyData))
```

```
}

/** Расписание группы по номеру группы **/
fun getGroup(number: String): ScheduleGroup? {
    for (group in scheduleGroup)
        if ( group.number == number )
            return group
    return null
}

/** Расписание группы по id группы **/
fun getGroup(id: Int): ScheduleGroup? {
    for (group in scheduleGroup)
        if ( group.id == id )
            return group
    return null
}

/** название дня по id **/
fun getDayName(id: Int) = days[id]

/** возвращает список преподавателей подходящих под запрос **/
fun getListSimilarTeacher(request: String): MutableList<Teacher> {
    val teacherList =  mutableListOf<Teacher>()
    for ((_, value) in teacher ){
        if ( value.name.contains(request) )
            teacherList.add(value)
    }
    return teacherList
}

/** удаляет пару из расписания группы по ссылке **/
fun deleteLessonForGroup(link: Link) {
    logger.info { "deleteLessonForGroup $link" }
    for ( group in scheduleGroup )
        if ( group.number == this.group[link.viewMainId] )
            for ( day in group.day )
                if ( day.nameId == link.viewDay)
                    for ( pair in day.pair )
                        if ( pair.timeId == link.viewTime )
                            pair.let {
                                when (link.viewTypeDay) {
```

```
                        PAIR_TIME.EQUALLY -> it.high = null
                        PAIR_TIME.HIGH -> it.high = null
                        PAIR_TIME.LOW -> it.low = null
                    }
                }
    }

    /** подсчитывает количество преподавателей подъходящих под запрос **/
    fun calcNumberOfSuitableTeachers(request: String): Int {
        var teacherCounter
= 0
        for ((_, value) in teacher ){
            if ( value.name.contains(request) )
                teacherCounter++
        }
        return teacherCounter
    }

    /** возвращает объект времени по id **/
    fun getTime(id: Int) = timeTable[id]

    fun getTeacherName(id: Int) = teacher[id]?.name ?: ""

    fun getLessonName(id: Int) = lessonName[id]

    fun getBuilding(id: Int) = building[id].toString()

    fun getClassroom(id: Int) = classroom[id].toString()

    fun getGroupNumber(id: Int) = group[id].toString()

    /** возвращает id по имени дня **/
    fun getDayId(name : String) : Int {
        for ( dayNameId in 0 until days.size ) {
            if ( days[dayNameId] == name )
                return dayNameId
        }
        return -1
    }

    private fun autoFillDays() {
        logger.info { "autoFillDays" }
```

```
        days.add("Вне сетки расписания")
        days.add("Понедельник")
        days.add("Вторник")
        days.add("Среда")
        days.add("Четверг")
        days.add("Пятница")
        days.add("Суббота")
    }

    /** преобразование html представления MAP в объект **/
    private fun String.fillingMap(map : MutableMap<Int, String>) {
        if (this.length >= 6) {
            val key = this.substringAfter("value=\"")
                    .substringBefore("\">").toInt()
            val value = this.substringAfter("""">""")
                     .substringBefore("""</option>\r\n\r\n""")
            map[key] = value
        }
    }

    /** добавление ГРУППЫ **/
    fun add(group: ScheduleGroup)
    = scheduleGroup.add(group)

    /** id преподавателя по его имени **/
    fun getTeacherId(string: String): Int {
        for ((key,value ) in teacher)
            if ( value.name == string )
                return key
        return 0
    }

    /** Получение id АУДИТОРИИ **/
    fun getClassroomId(classroomName: String): Int {
        for ((key,value ) in classroom)
            if ( value == classroomName )
                return key
        // Странная непонятная ситуация, но вдруг занятие проходит в аудитории которой нет в
списке
        // как оказалось вообще обыденность
        // -1 это тоже корректное значение
        logger.warn { "classroom not find" }
```

```
    return -1
}

/** Получение id ЗДАНИЯ **/
fun getBuildingId(string: String): Int {
    for ((key,value) in building)
        if ( value == string )
            return key
    return -1
}

/** возвращает id по имени УРОКА **/
fun getLessonId(name: String): Int {
    for ( i in 0 until lessonName.size)
        if ( lessonName[i] == name )
            return i
    lessonName.add(name)
    return lessonName.size - 1  // Если предмет бы добавленн его id будет последним
}

/** Получение id ВРЕМЕНИ и добавления его в базу при отсутствии **/
fun getTimeId(string: String): Int {
    val name_ = string.substringBefore(" (")
    for ( i in 0 until timeTable.size) {                    // Поиск наличия такой записи
        if ( timeTable[i].name == name_ )
            return i
    }
    if ( """[\d-]""".toRegex().containsMatchIn(string)  ) {  // Есть время
        val strTimeFrom = string.substringAfter("(").substringBefore("–")
        val strTimeTo   = string.substringAfter("–").substringBefore(")")
        logger.info { "getTimeId time [$string] [$name_] [$strTimeFrom] [$strTimeTo]" }
        timeTable.add(
            TimeTable(
                name_,
                time(strTimeFrom),
                time(strTimeTo)
            )
        )
    } else {                                            // Время не указанно
        logger.info { "getTimeId [$string] [$name_]" }
        timeTable.add(
            TimeTable(
```

```
                name_,
                null,
                null
            )
        )
    }
    return timeTable.size - 1
}

/** преобразования строки со временем в объект **/
private fun time(html: String) = Time(
    html.substringAfter("(").substringBefore(":").toInt(),
    html.substringAfter(":").substringBefore(")").toInt(),
)

/** Получение текста расписания ГРУППЫ, преобразование в объект и добавления его в базу **/
fun groupFromText(text: String) =
    text.split("</option>").forEach { it.fillingMap(group) }

/** Получение текста расписания ПРЕПОДАВАТЕЛЯ, преобразование в объект и добавления его в базу **/
fun teacherFromText(text: String) =
    text.split("</option>").forEach {
        if ( it.length >= 6 ) {
            val key = it.substringAfter("value=\"").substringBefore("\">").toInt()
            val value = it.substringAfter("""">""").substringBefore("""</option>\r\n\r\n""")
            val nameId = value.substringBefore(""" - """)
            val position = value.substringAfter(""" - """)
            teacher[key] = Teacher(nameId, position)
        }
    }

/** Получение текста о АУДИТОРИЯХ, преобразование в объект и добавления его в базу **/
fun classroomFromText(html: String) =
    html.split("</option>").forEach { it.fillingMap(classroom) }

/** Получение текста о ЗДАНИЯХ, преобразование в объект и добавления его в базу **/
fun buildingFromText(html: String) =
    html.split("</option>").forEach { it.fillingMap(building) }

/** -1 если не найдено **/
```

```
    fun getGroupId(number: String) : Int {
        for((key, value) in group)
            if ( value == number )
                return key
        return -1
    }

    /** расписание преподавателя по его id **/
    fun getTeacher(id: Int): ScheduleTeacher? {
        for (teacher in scheduleTeacher)
            if ( teacher.id == id )
                return teacher
        return null
    }

    override fun toString() =
        """
        group :
        $group
        teacher :
        $teacher
        classroom :
        $classroom
        building :
        $building
        """.trimIndent()
}

// @filename /core/data/LessonType.kt
package org.suai.schedule.core.data

/*
data class LessonType(
    /** сокращение (ЛР, Л) **/
    val liter: String,
    /** расшифровка **/
    val name: String
) {
    override fun toString(): String {
        return """
            ( $liter - $name )
        """.trimIndent()
```

```
    }
}
*/

// @filename /core/data/OnlyData.kt
package org.suai.schedule.core.data

open class OnlyData {
    /** Строка с номером сборки и даты **/
    open var time = ""

    /** Map преподавателей по id **/ open var teacher    = mutableMapOf<Int, Teacher>()
    /** Map аудиторий по id      **/ open var classroom  = mutableMapOf<Int, String>()
    /** Map сданий по id         **/ open var building   = mutableMapOf<Int, String>()
    /** Map груп по id           **/ open var group      = mutableMapOf<Int, String>()

    /** Список имён и времени пар **/       open var timeTable  = mutableListOf<TimeTable>()
    /** Список названий предметов **/       open var lessonName = mutableListOf<String>()
    /** Список имён дней (автогенерация) **/open var days       = mutableListOf<String>()

    fun toData(data : Data) {
        data.teacher    = this.teacher
        data.classroom  = this.classroom
        data.building   = this.building
        data.group      = this.group

        data.timeTable  = this.timeTable
        data.lessonName = this.lessonName
        data.days       = this.days
    }
}

// @filename /core/data/Teacher.kt
package org.suai.schedule.core.data

data class Teacher (
    val name: String,
    /** должность **/
    val position: String
) {
    override fun toString(): String {
        return """
```

```
            ( $name - $position )
        """.trimIndent()
    }
}

// @filename /core/data/Time.kt
package org.suai.schedule.core.data

data class Time(
    var hours: Int,
    var min: Int
) {
    override fun toString() = "$hours:${if(min==0) "00" else min}"
}

// @filename /core/data/TimeTable.kt
package org.suai.schedule.core.data

data class TimeTable(
    val name:     String,
    var timeFrom: Time?,    // Если null, вместо времени дублируеться имя
    var timeTo:   Time?
)


// @filename /core/manager/findManager.kt
package org.suai.schedule.core.manager

import androidx.compose.runtime.MutableState
import mu.KotlinLogging
import org.suai.schedule.core.data.Data
import org.suai.schedule.core.net.Net
import org.suai.schedule.model.Dir
import org.suai.schedule.model.schedule.ScheduleGroup
import org.suai.schedule.utils.file.checkIfFileExists
import org.suai.schedule.view.*
import org.suai.schedule.view.logger

object findManager {
    private val logger = KotlinLogging.logger {}

    /** загружает данные расписания с использованием кэша **/
```

```
    fun findGroup(number: String, ShearId: MutableState<Int>, type: MutableState<FIND>) {
        val id = Data.getGroupId(number)
        if ( id != -1 ) {           // Проверка знаем ли такую группу
            if ( Data.getGroup(id) == null ) {          // Проверка наличия в памяти
                val fileName = Dir.getScheduleCacheName(number, FIND.GROUP)         // Получение
имени файда
                if ( checkIfFileExists(Dir.cache, fileName) ) {     // Проверка, есть ли данная группа в
кэше
                    val fullGroup = loadScheduleFromCache(fileName, ScheduleGroup())   // Загруска из
кэша
                    if ( fullGroup is ScheduleGroup) {
                        Data.add(fullGroup)
                        val id = Data.getGroup(number)?.id
                        if ( id != null ) {
                            type.value = FIND.GROUP
                            ShearId.value = id
                        }
                        else
                            logger.error { "Error add group from fullGroup" }


                    } else {
                        logger.error { "Error to reading group id=[$id] from cache" }
                        Net.loadGroup(id, ShearId, type)
                    }
                } else {
                    Net.loadGroup(id, ShearId, type)            // Запрос
                }
            } // else - группа находиться в памяти
        }
        else
            logger.error { "findForGroup group not found" }
    }

    /** обработка строки и выполнение запроса **/
    fun processingString(mString: MutableState<String>, id: MutableState<Int>, type:
MutableState<FIND>) {
        val string = mString.value
        if ( string.isNotEmpty() ) {
            if (string[0] == '!' || string[0] == '?')
                commandExecutive(string)
            else {
                when (val determine = DetermineSearchType(string)) {
```

```
                FIND.NOPE       -> {
                    findNope(id, type)
                }
                FIND.GROUP      -> {
                    findGroup(string, id, type)
                }
                FIND.TEACHER    -> {
                    val count   = Data.calcNumberOfSuitableTeachers(string)
                    val similar = Data.getListSimilarTeacher(string)
                    logger.info { "DetermineSearch $determine [$count] [$similar]" }
                    if ( count == 1 ) {
                        findTeacher(similar[0].name, id, type)
                    }
                }
                FIND.CLASSROOM  -> {
                    findClassroom(string)
                }
                FIND.CROSSING_WITH_TEACHER -> {
                    findClassroomWithTeacher(string)
                }
            }
        }
    }
    else
        findNope(id, type)
  }

}

// @filename /core/manager/htmlManager.kt
package org.suai.schedule.core.manager

import mu.KotlinLogging
import org.suai.schedule.core.data.Data
import org.suai.schedule.core.data.Data.getClassroomId
import org.suai.schedule.core.data.Data.getTeacherId
import org.suai.schedule.model.schedule.*

object htmlManager {
  private val logger = KotlinLogging.logger {}

  fun fillTeacher(html: String) =
```

```
    if ( Data.checkActual(html) ) {
        val number = html.substringAfter("Расписание для преподавателя - ")
            .substringBefore("</h2>")
        val htmlSchedule = html.substringAfter("""</h2>""")
            .substringBefore("</div></div></div></div>")
        logger.info { "fillTeacher \n$html" }
        // преобразование html представления РАСПИСАНИЯ ГРУППЫ в объект и сохраняет его
        scheduleFromText(htmlSchedule, number, ScheduleTeacher())
        true
    } else {
        logger.error { "loadGroup request error" }
        false
    }

fun fillGroup(html: String) =
    if ( Data.checkActual(html) ) {
    val number = html.substringAfter("Расписание для группы - ")
                .substringBefore("</h2>")
    val htmlSchedule = html.substringAfter("""</h2>""")
                    .substringBefore("</div></div></div></div>")
    logger.info { "forGroup \n$html" }
    // преобразование html представления РАСПИСАНИЯ ГРУППЫ в объект и сохраняет его
    scheduleFromText(htmlSchedule, number, ScheduleGroup())
    true
} else {
    logger.error { "loadGroup request error" }
    false
}

/** Функция передаваемая лямбдой (вынести в логику) **/
fun fillFirstLoad(string: String?) =
    if ( string != null && string.isNotEmpty() ) {
        val htmlTextGroup = string.substringAfter("""ctl05">""")
                        .substringBefore("</select>")
        logger.info {"groupId $htmlTextGroup"}       // <option value="12">1010М</option>
        Data.groupFromText(htmlTextGroup)

        val htmlTextTeacher = string.substringAfter("""ctl06">""")
                        .substringBefore("</select>")
        logger.info {"teacher $htmlTextTeacher"}     // <option value="95">Авдеев В.А. - доцент,
к.т.н.</option>
        Data.teacherFromText(htmlTextTeacher)
```

```
        val htmlTextClassroom = string.substringAfter("""ctl08">""")
                                .substringBefore("</select>")
        logger.info {"classroom $htmlTextClassroom"} // <option value="55">11-01a</option>
        Data.classroomFromText(htmlTextClassroom)

        val htmlTextBuilding = string.substringAfter("""ctl07">""")
                               .substringBefore("</select>")
        logger.info {"building $htmlTextBuilding"}   // <option value="1">Б.Морская 67</option>
        Data.buildingFromText(htmlTextBuilding)

        // Заполнение даты, с игнорированием сравнения
        Data.time = string.substringAfter("(Сборка: ")
                .substringBefore(")")
        true
    } else {
        logger.error { "firstLoad wrong answer" }
        false
    }

/** функция для заполнения поля "вне сетки расписания" **/
fun fillOutOfSchedule(pairListStringHtml: String) : String{
    TODO()
}

/** общая функция для заполнения расписаний **/
private fun scheduleFromText(html: String, name: String, schedule: Schedule) {
    when (schedule) {
        is ScheduleGroup -> {
            schedule.number = name
            schedule.id     = Data.getGroupId(name)
        }
        is ScheduleTeacher -> {
            schedule.name = name
            schedule.id = getTeacherId(name.substringBefore(" - "))
        }
        is ScheduleClassroom -> {
            // not testing
            schedule.number = name
            schedule.id = getClassroomId(name)
        }
    }
```

```
    if ( schedule.id == -1 )
        logger.error { "scheduleFromText schedule id = -1 for [$name]" }
        html.split("<h3>").forEach {
            val day = dayFromText(it)
            day?.let { Day -> schedule.day.add(Day) }
        }
    if (Data.addSchedule(schedule))
        logger.error { "error add new schedule to data" }
    logger.info { schedule.toString() }
}

/** преобразование html представления ДНЯ в объект **/
private fun dayFromText(html: String) : Day? {
    if ( html.isEmpty() ) {
        logger.warn { "dayFromText text is empty " }
        return null
    }
    logger.info { "dayFromText $this" }
    val name = html.substringAfter("""<h3>""")
                .substringBefore("""</h3>""")
    val nameId = Data.getDayId(name)
    if ( nameId == -1 ) {
        logger.error { "dayFromText Non-existent day[$name]" }
        return null
    }
    val day = Day(nameId)
    html.substringAfter("<h4>")
        .split("<h4>")
        .forEach {
        day.pair.add(pairFromText(it))
    }
    return day
}

/** преобразование html представления ПАРЫ в объект **/
private fun pairFromText(html: String) : Pair {
    logger.info { "lessonFromText [$html]" }
    val pair = Pair()
    pair.timeId = Data.getTimeId(
        html.substringAfter("<h4>")
            .substringBefore("""</h4>""")  // Дата
    )
```

```
    val flagUp = """(class="up")""".toRegex().containsMatchIn(html)
    val flagDn = """(class="dn")""".toRegex().containsMatchIn(html)
    if ( flagUp && flagDn ) {                    // Мигалка
        pair.equally = false
        pair.high =
            fillLessonFromString(
                html.substringBefore("""<div class="study"><span><b class="dn"""")
            )
        pair.low  = fillLessonFromString(
            html.substringAfter("""</a></span></div>""")
        )
        logger.info { "lessonFromText two [${pair.high}] [${pair.low}]" }
    } else if ( !flagUp && !flagDn ) {           // Одинаковое
        pair.equally = true
        pair.high = fillLessonFromString(html)
        logger.info { "lessonFromText one [${pair.high}]" }
    } else if ( flagUp ) {                       // Только верхняя
        pair.equally = false
        pair.high = fillLessonFromString(html)
        logger.info { "lessonFromText up [${pair.high}]" }
    } else {                                     // только нижняя
        pair.equally = false
        pair.low = fillLessonFromString(html)
        logger.info { "lessonFromText down [${pair.low}]" }
    }
    return pair
}

/** преобразование html представления УРОКА в объект **/
private fun fillLessonFromString(html: String) : Lesson {
    val lesson = Lesson(
        nameId = Data.getLessonId(
            html.substringAfter("</b> &ndash; ")
                .substringBefore("""" <em> &ndash;""")
        ),
        type = getType(
            html.substringAfter("<b>")
                .substringBefore("</b>")
        ),
        buildingId = Data.getBuildingId(
            html.substringAfter("<em> &ndash; ")
                .substringBefore(", ауд.")
```

```
            ),
        )
        if ( """Преподаватель: <a href=""".toRegex().containsMatchIn(html) )            // Проверка на наличие преподавателя
            lesson.teacherId.add(
                html.substringAfter("Преподаватель: <a href=\"?p=")
                    .substringBefore("\">")
                    .toInt())
        else if ("""Преподаватели: <a href=""".toRegex().containsMatchIn(html) )        // Проверка на нескольких преподавателей
            html.substringAfter("Преподаватели: <a href=\"?p=")
                .substringBefore("</a></span>").split("?p=")


       .forEach {
                lesson.teacherId.add(it.substringBefore("\">").toInt())
            }
        html.substringAfter("ауд. ")
            .substringBefore("</em></span>")
            .split("; ")
            .forEach {
            lesson.classroomId.add(Data.getClassroomId(it))
        }
        if ( "Группа: ".toRegex().containsMatchIn(html) )              // Одна группа
            lesson.groupId.add(
                html.substringAfter("<a href=\"?g=")
                    .substringBefore("\">")
                    .toInt())
        else                                                  // Много групп
            html.substringAfter("""Группы: <a href="?g=""")
                .split("?g=")
                .forEach {
                lesson.groupId.add(
                    it.substringBefore("\">")
                    .toInt())
            }
        return lesson
    }
}


// @filename /core/net/Net.kt
package org.suai.schedule.core.net
```

```
import androidx.compose.runtime.MutableState
import io.ktor.client.request.*
import mu.KotlinLogging
import org.suai.schedule.core.data.Data
import org.suai.schedule.core.data.Data.buildingFromText
import org.suai.schedule.core.data.Data.classroomFromText
import org.suai.schedule.core.manager.htmlManager
import org.suai.schedule.model.Dir
import org.suai.schedule.model.schedule.ScheduleGroup
import org.suai.schedule.utils.file.checkIfFileExists
import org.suai.schedule.utils.file.writeFile
import org.suai.schedule.utils.json.toJson
import org.suai.schedule.utils.ktorHttpClient
import org.suai.schedule.utils.runBlocking
import org.suai.schedule.view.FIND

object Net {
    private val logger = KotlinLogging.logger {}

    private val mainUrlSchedule = "https://rasp.guap.ru/"           // семест
    private val mainUrlExamination = "https://raspsess.guap.ru/"    // экзамены

    fun get(httpsURL: String): MutableList<String> {
        return runBlocking {
            val content = ktorHttpClient.get<String>(httpsURL)
            content.lines().toMutableList()
        }
    }

    /** ассинхронный get запроса, выполняющий лямбду **/
    private fun load(url: String, f: (String) -> Unit) {
        val textRequest = get(url)
        f(textRequest.toString())
    }

    /** первичная загруска списка груп и преподавателей, зданий и аудиторий **/
    fun firstLoad() =
        load(mainUrlSchedule) { textRequest: String? -> htmlManager.fillFirstLoad(textRequest) }

    /** загрузка расписания преподавателя по id **/
    fun loadTeacher(id: Int, ShearId: MutableState<Int>, type: MutableState<FIND>) {
```

```
        if ( id == -1)
            logger.error { "load teacher with id -1" }       // хз почему не else
        logger.info { "load for teacher with id $id " }
        load("$mainUrlSchedule?p=$id") {
            textRequest: String -> htmlManager.fillTeacher(textRequest)
            type.value    = FIND.TEACHER
            ShearId.value = id
        }
    }

    /** загрузка расписания группы по id **/
    fun loadGroup(id: Int, ShearId: MutableState<Int>, type: MutableState<FIND>) {
        if ( id == -1 )                             // Группа не существует
            logger.error { "load group with id -1" }
        logger.info { "load for group with id $id " }
        load("$mainUrlSchedule?g=$id") {              // Загруска
            textRequest: String -> htmlManager.fillGroup(textRequest)
            type.value    = FIND.GROUP
            ShearId.value = id
    // В надежде что этим займёться другой поток
            val group = Data.getGroup(id)
            if ( group is ScheduleGroup ) {
                Data.add(group)
                val fileName: String = Dir.getScheduleCacheName(group)
                if ( !checkIfFileExists(Dir.cache, fileName) ) {       // Если файл кэша отсутствует
                    //writeFile(Dir.cache, fileName, toJson(group.toFull()))
                    //writeFile(Dir.cache, "id_$fileName", toJson(group))
                }
            }
            else
                logger.error { "group not found, but he fill " }
        }
    }

    /** заполнение данных о преподавателе **/
    //private fun fillTeacher(string: String?) {}
    /*
        if ( string != null && Data.checkActual(string) ) {

            val name = string.substringAfter("Расписание для преподавателя -
").substringBefore("</h2>")
            val html = string.substringAfter("""</h2>""").substringBefore("</div></div></div></div>")
```

```
            logger.info { "fillTeacher \n$html" }
            //Data.scheduleFromTextForTeacher(html,  ) // передаю текст вего расписания
            true
        } else {
            logger.error { "loadForGroup request error" }
            false
        }*/

    /*
    /** заполнение данных о группе **/
    private fun fillGroup(string: String?) =
        if ( string != null && Data.checkActual(string) ) {
            val number = string.substringAfter("Расписание для группы - ").substringBefore("</h2>")
            val html = string.substringAfter("""</h2>""").substringBefore("</div></div></div></div>")
            logger.info { "fillGroup \n$html" }
            Data.Group(number)
            true
        } else {
            logger.error { "loadGroup request error" }
            false
        }
     */
}

// @filename /core/settings/LongTimeSettings.kt
package org.suai.schedule.core.settings

import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember

/** Параметры которые остаються после перезапуска **/
open class  LongTimeSettings {
    lateinit var theme      : MutableState<Int>
    lateinit var smartFind  : MutableState<Boolean>
    lateinit var hideGroup  : MutableState<Boolean>
    lateinit var hideTeacher: MutableState<Boolean>

    @Composable
    fun initMain() {
        /** ID темы **/
```

```
        theme = remember { mutableStateOf(0) }
        /** Умный поиск без слайдер выбора критерия оиска (уризает функционал) */
        smartFind = remember { mutableStateOf(false) }
        /** Скрыть группы в отображении **/
        hideGroup = remember { mutableStateOf(false) }
        /** Скрыть преподавателей в отображении**/
        hideTeacher = remember { mutableStateOf(false) }
    }

    fun toSettings(settings: Settings) {
        settings.theme          = theme
        settings.smartFind      = smartFind
        settings.hideGroup      = hideGroup
        settings.hideTeacher    = hideTeacher
    }
}

// @filename /core/settings/Settings.kt
package org.suai.schedule.core.settings

import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import org.suai.schedule.model.Link

object Settings : LongTimeSettings() {
    lateinit var advanceFindFlag : MutableState<Boolean>
    lateinit var loading         : MutableState<Boolean>
    lateinit var linkToSelect    : MutableState<Link?>

    @Composable
    fun init() {
        initMain()
        advanceFindFlag = remember { mutableStateOf(false) }    // необходимо дополнитеьно меню
        loading         = remember { mutableStateOf(false) }    // Индикция загруски
        linkToSelect    = remember { mutableStateOf(null) }     // Ссылка на выделенный элемент
    }
}


// @filename /model/Dir.kt
```

```
package org.suai.schedule.model

import mu.KotlinLogging
import org.suai.schedule.model.schedule.ScheduleClassroom
import org.suai.schedule.model.schedule.ScheduleGroup
import org.suai.schedule.model.schedule.Schedule
import org.suai.schedule.model.schedule.ScheduleTeacher
import org.suai.schedule.view.FIND

object Dir {
    lateinit var cache: String
    lateinit var data : String

    fun init(
        cache: String = "./cache/",
        data : String = "./data/"
    ) {
        this.cache = cache
        this.data  = data
    }

    // Можно добавить возможность записи в отдельные директории
    /** возвращает имя для файла в кэше **/
    fun getScheduleCacheName(schedule : Schedule) =
        when (schedule) {
            is ScheduleGroup       -> "g_${schedule.number}.json"
            is ScheduleTeacher     -> "t_${schedule.name}.json"
            is ScheduleClassroom   -> "c_${schedule.number}.json"
            else -> ""
        }

    /** возвращает имя для файла в кэше ( name = number ) **/
    fun getScheduleCacheName(name: String, type: FIND) =
        when ( type ) {
            FIND.GROUP      -> "g_$name.json"
            FIND.TEACHER    -> "t_$name.json"
            FIND.CLASSROOM  -> "c_$name.json"
            else -> {
                KotlinLogging.logger {}.error { "getScheduleCacheName FIND not processing" }
                ""
            }
        }
```

```
 }

// @filename /model/Link.kt
package org.suai.schedule.model

import org.suai.schedule.model.schedule.PAIR_TIME
import org.suai.schedule.view.FIND

data class Link(
    var viewType: FIND = FIND.NOPE,
    var viewMainId: Int = 0,
    var viewDay: Int = 0,
    var viewTime: Int = 0,
    var viewTypeDay: PAIR_TIME = PAIR_TIME.EQUALLY
){
    /** Возвращает, копию ссылки **/
    fun getCopy() = Link(viewType,viewMainId,viewDay,viewTime,viewTypeDay)
}

// @filename /model/Theme.kt
package org.suai.schedule.model

import androidx.compose.ui.graphics.Color

data class Theme(
    val id: Int,
    val name: String,
    val background: Color,
    val first: Color,
    val second: Color,
    val lowLesson: Color,
    val topLesson: Color,
    val lesson: Color,
    val day: Color,
    val font: Color,
    val sabFont: Color
)

// @filename /model/schedule/Day.kt
package org.suai.schedule.model.schedule

import org.suai.schedule.core.data.Data
```

```
import org.suai.schedule.model.schedule.full.fullDayModel

/** Один день, одной группы **/
data class Day(
    /** ID дня **/


 val nameId: Int = 0
) {
    /** Список занятий на день **/
    var pair = mutableListOf<Pair>()

    fun toFull() : fullDayModel{
        val fullDay = fullDayModel(Data.getDayName(nameId))
        pair.forEach { fullDay.pair.add( it.toFull() ) }
        return fullDay
    }
}

// @filename /model/schedule/Lesson.kt
package org.suai.schedule.model.schedule

import org.suai.schedule.core.data.Data
import org.suai.schedule.model.schedule.full.fullLessonModel

/** Модель для хранения данных об одном занятии **/
data class Lesson(
    /** ID предмета **/
    var nameId      :  Int = 0,
    /** ID типа занятия     [Л,ПР] **/
    var type: LESSON_TYPE,
    /** ID здания           [БМ, Гаст, Ленса] **/
    var buildingId  :  Int = 0,
) {
    /** ID аудитории **/
    val classroomId = mutableListOf<Int>()
    /** ID преподавателя **/
    val teacherId   = mutableListOf<Int>()
    /** ID групп на занятии (все) **/
    val groupId     = mutableListOf<Int>()

    fun toFull(): fullLessonModel {
        val lesson = fullLessonModel(
```

```
            Data.getLessonName(nameId),
            type = type.liter,
            Data.getBuilding(buildingId)
        )
        classroomId.forEach { lesson.classroom.add  ( Data.getClassroom(it)   ) }
        teacherId.forEach   { lesson.teacherName.add( Data.getTeacherName(it) ) }
        groupId.forEach     { lesson.groupNumber.add( Data.getGroupNumber(it) ) }
        return lesson
    }
}


// @filename /model/schedule/LESSON_TYPE.kt
package org.suai.schedule.model.schedule

/**@param L  лекция
 * @param PR практическое занятие или семинар
 * @param LR лабораторные занятия
 * @param KP курсовой проект
 * @param KR курсовая работа **/
sealed class LESSON_TYPE(open val liter: String, open val name: String){
    override fun toString(): String {
        return """
            ( $liter - $name )
        """.trimIndent()
    }
}

data class LT_Lecture(override val liter: String = "Л", override val name: String = "лекция") :
LESSON_TYPE("","лекция")
data class LT_Practical(override val liter: String = "ПР", override val name: String = "практическое
занятие или семинар") : LESSON_TYPE("", "")
data class LT_Laboratory(override val liter: String = "ЛР", override val name: String = "лабораторные
занятия") : LESSON_TYPE("", "")
data class LT_CourseProject(override val liter: String = "КП", override val name: String = "курсовой
проект") : LESSON_TYPE("", "")
data class LT_CourseWork(override val liter: String = "КР", override val name: String = "курсовая
работа") : LESSON_TYPE("", "")

fun getType(liter: String) : LESSON_TYPE {
    return when (liter) {
        "Л" -> return LT_Lecture()
```

```
        "ПР" -> return LT_Practical()
        "ЛР" -> return LT_Laboratory()
        "ЛР" -> return LT_CourseProject()
        "КР" -> return LT_CourseWork()
        else -> LT_Lecture()
    }
}

// @filename /model/schedule/Pair.kt
package org.suai.schedule.model.schedule

import org.suai.schedule.core.data.Data
import org.suai.schedule.model.schedule.full.fullPairModel

enum class PAIR_TIME { HIGH, LOW, EQUALLY }

/** Одна пара (по времени) **/
data class Pair(
    /** ID времени **/
    var timeId: Int         = 0,
    /** true - пара одинаковая для двух недель, смотреть по high **/
    var equally : Boolean   = false,
    /** Верхняя неделя **/
    var high: Lesson?  = null,
    /** Нижняя неделя **/
    var low: Lesson?   = null
) {

    fun toFull(): fullPairModel {
        val pairTime = Data.getTime(timeId)
        return fullPairModel(
            pairTime.name,
            pairTime.timeFrom,
            pairTime.timeTo,
            equally,
            high?.toFull(),
            low?.toFull()
        )
    }
}

// @filename /model/schedule/Schedule.kt
```

```
package org.suai.schedule.model.schedule

sealed class Schedule {
    /** ID группы/преподавателя/аудитории **/
    open var id  = 0
    /** Список дней в недели **/
    open var day = mutableListOf<Day>()
}

class ScheduleClassroom : Schedule() {
    /** Номер аудитории **/
    var number  = ""
}

class ScheduleGroup : Schedule() {
    /** Номер группы **/
    var number  = ""
}

class ScheduleTeacher : Schedule() {
    var name: String = ""
}

// @filename /model/schedule/full/fullClassroomScheduleModel.kt
package org.suai.schedule.model.schedule.full

class fullClassroomScheduleModel : fullScheduleModel()  {
    /** Номер аудитории **/
    var number  = ""
}

// @filename /model/schedule/full/fullDayModel.kt
package org.suai.schedule.model.schedule.full

import org.suai.schedule.core.data.Data

data class fullDayModel(
    /** Название дня **/
    val name: String = ""
) {
    /** Список занятий на день **/
    var pair = mutableListOf<fullPairModel>()
```

```
}

// @filename /model/schedule/full/fullGroupScheduleModel.kt
package org.suai.schedule.model.schedule.full

import org.suai.schedule.core.data.Data
class fullGroupScheduleModel : fullScheduleModel() {
    /** Номер группы **/
    var number  = ""
}

// @filename /model/schedule/full/fullLessonModel.kt
package org.suai.schedule.model.schedule.full

data class fullLessonModel(
    /** Название предмета **/
    var name      : String = "",
    /** Литера типа занятия     [Л,ПР] **/
    var type      : String = "",
    /** Здание здания           [БМ, Гаст, Ленса] **/
    var building  : String = "",
) {
    /** Номер аудитории **/
    val classroom   = mutableListOf<String>()
    /** Имя преподавателя **/
    val teacherName = mutableListOf<String>()
    /** Номера групп на занятии (все) **/
    val groupNumber = mutableListOf<String>()
}

// @filename /model/schedule/full/fullPairModel.kt
package org.suai.schedule.model.schedule.full

import org.suai.schedule.core.data.Time

data class fullPairModel (
    /** Название/номер пары **/
    var time: String        = "",
    /** Время начала пары **/
    var timeStart: Time? = Time(0, 0),
    /** Время окончания пары **/
    var timeEnd: Time? = Time(0, 0),
```

```
    /** true - пара одинаковая для двух недель, смотреть по high **/
    var equally : Boolean   = false,
    /** Верхняя неделя **/
    var high: fullLessonModel?  = null,
    /** Нижняя неделя **/
    var low: fullLessonModel?   = null) {

}

// @filename /model/schedule/full/fullScheduleModel.kt
package org.suai.schedule.model.schedule.full


open class fullScheduleModel {
    /** Список дней в недели **/
    var day = mutableListOf<fullDayModel>()

}

// @filename /model/schedule/full/fullTeacherScheduleModel.kt
package org.suai.schedule.model.schedule.full

class fullTeacherScheduleModel: fullScheduleModel() {
    /** ФИО преподавателя **/
    var name: String = ""
}

// @filename /screen/Schedule/ScheduleScreen.kt
package org.suai.schedule.view

import androidx.compose.foundation.layout.Column
import androidx.compose.material.Scaffold
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import org.suai.schedule.core.Themes
import org.suai.schedule.core.manager.findManager
import org.suai.schedule.core.settings.Settings
import org.suai.schedule.model.Link
import org.suai.schedule.view.scheduleView.ScheduleBlockView
```

```
@Composable
fun ScheduleScreen() {
    val type                    = remember { mutableStateOf(FIND.NOPE) }
    val findString              = remember { mutableStateOf("") }
    val id                      = remember { mutableStateOf(0) }
    val flagPushingEnter        = remember { mutableStateOf(false) }
    val recompose               = remember { mutableStateOf(false) }
    val link: MutableState<Link?>   = remember { mutableStateOf(null) }
    val theme                   = remember { mutableStateOf(Themes.actual) }

    Scaffold(
        floatingActionButton = {
            //FloatingActionButton(onClick = {}) {
            //Text("X")
            //}
        },
        drawerContent = {
            AdvFindView(theme)
        },
        content = {
            if ( recompose.value )
                recompose.value = false
            else {
                Column {
                    link.value = Settings.linkToSelect.value
                    if (link.value != null) TopBarView(id, type, link, recompose, theme)
                    FindView(findString, flagPushingEnter, theme)
                    findManager.processingString(findString, id, type)
                    ScheduleBlockView(id, type, theme)
                }
            }
        },
    )
}

// @filename /style/Palette.kt
package org.suai.schedule.style

import androidx.compose.ui.graphics.Color

val DarkGreen = Color(16, 139, 102)
val Gray = Color.DarkGray
```

```
val LightGray = Color(100, 100, 100)
val DarkGray = Color(32, 32, 32)
val PreviewImageAreaHoverColor = Color(45, 45, 45)
val ToastBackground = Color(23, 23, 23)
val MiniatureColor = Color(50, 50, 50)
val MiniatureHoverColor = Color(55, 55, 55)
val Foreground = Color(210, 210, 210)
val TranslucentBlack = Color(0, 0, 0, 60)
val TranslucentWhite = Color(255, 255, 255, 20)
val Transparent = Color.Transparent

// @filename /style/TextViews.kt
package org.suai.schedule.style

import androidx.compose.foundation.layout.Column
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

@Composable
fun textExample() {
   val str = "Съешь еще этих мягких французских булок"
   Column {
      Subtitle1(str, Color.Black)
      Subtitle2(str, Color.Black)
      Subtitle3(str, Color.Black)
      Subtitle4(str, Color.Black)
   }
}

@Composable
fun Subtitle1(text: String, color: Color, modifier: Modifier = Modifier) {
   Text(
      text = text,
      modifier = modifier,
      style =TextStyle(  color = color,
```

```
                fontSize = 16.sp,
                fontWeight = FontWeight.Normal)
    )
}

@Composable
fun Subtitle2( text: String, color: Color, modifier: Modifier = Modifier) {
    Text(
        text = text,
        modifier = modifier,
        style =TextStyle(  color = color,
            fontSize = 14.sp,
            fontWeight = FontWeight.Normal)
    )
}

@Composable
fun Subtitle3( text: String, color: Color, modifier: Modifier = Modifier) {
    Text(
        text = text,
        modifier = modifier,
        style =TextStyle(  color = color,
            fontSize = 24.sp,
            fontWeight = FontWeight.Bold)
    )
}

@Composable
fun Subtitle4( text: String, color: Color, modifier: Modifier = Modifier) {
    Text(
        text = text,
        modifier = modifier,
        style =TextStyle(  color = color,
            fontSize = 18.sp,
            fontWeight = FontWeight.Medium)
    )
}

// @filename /utils/Coroutines.kt
package org.suai.schedule.utils

import kotlinx.coroutines.CoroutineScope
```

```
import kotlin.coroutines.CoroutineContext
import kotlin.coroutines.EmptyCoroutineContext

expect fun <T> runBlocking(context: CoroutineContext = EmptyCoroutineContext, block: suspend
CoroutineScope.() -> T): T

// @filename /utils/Network.kt
package org.suai.schedule.utils

import io.ktor.client.*

val ktorHttpClient = HttpClient {}


// @filename /utils/platforn.kt
package org.suai.schedule.utils

/** возвращает строку с названием платформы **/
expect fun getPlatform(): String

// @filename /utils/file/checkIfFileExists.kt
package org.suai.schedule.utils.file

/** true - файл существует **/
expect fun checkIfFileExists(dir: String, source: String) : Boolean

// @filename /utils/file/readFile.kt
package org.suai.schedule.utils.file

/** чтение данных из файла (место, имя файла) **/
expect fun readFile(dir: String, source: String) : String

// @filename /utils/file/writeFile.kt
package org.suai.schedule.utils.file

/** запись данных в файл (место, имя файла, содержимое) **/
expect fun writeFile(dir : String, source: String, content: String)

// @filename /utils/json/fromJson.kt
package org.suai.schedule.utils.json

/** получение объкекта из json строки **/
```

```
expect fun fromJson(string: String, objectType: Any): Any?

// @filename /utils/json/toJson.kt
package org.suai.schedule.utils.json

/** преобразование объекта в json стоку **/
expect fun toJson(any: Any): String

// @filename /view/AdvFindView.kt
package org.suai.schedule.view

import androidx.compose.foundation.layout.padding
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp
import org.suai.schedule.model.Theme

@Composable
fun AdvFindView(theme: MutableState<Theme>) {
   logger.info { "recompose AdvFindView" }
   val action = remember { mutableStateOf(0) }
   SliderView(
      noActiveColor   = Color(256,100,100,0xFF),
      activeColor     = Color(100,256,100,100),
      active          = action,
      options         = listOf("Группы", "Преподаватель", "Аудитория" ),
      modifier        = Modifier.padding(horizontal = 3.dp),
      theme           = theme
   )
   logger.info { "AdvFindView slider select [${action.value}]" }
}

// @filename /view/FindView.kt
package org.suai.schedule.view

import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxWidth
```

```
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.text.selection.DisableSelection
import androidx.compose.material.OutlinedTextField
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import mu.KotlinLogging
import org.suai.schedule.core.data.Data
import org.suai.schedule.core.manager.findManager
import org.suai.schedule.core.net.Net.loadTeacher
import org.suai.schedule.model.Dir
import org.suai.schedule.model.Theme
import org.suai.schedule.utils.file.readFile
import org.suai.schedule.utils.json.fromJson

enum class FIND {
    GROUP, TEACHER, CLASSROOM, CROSSING_WITH_TEACHER, NOPE
}
internal val logger = KotlinLogging.logger{}

/** View поисковой строки **/
@Composable
fun FindView( findString: MutableState<String>, flagEnterPush: MutableState<Boolean>, theme:
MutableState<Theme>) {
    logger.info { "recompose findView" }
    Modifier.fillMaxWidth()
    Row {
        OutlinedTextField(
            value = findString.value,
            onValueChange = {
                if (it.isNotEmpty() && it[it.length - 1] == '\n') {
                    logger.info { "input enter" }
                    flagEnterPush.value = true
                }
                findString.value = it.replace("\n","")
```

```
        },
        label = {
            Text(
                text = "find",
                style = TextStyle(  color = Color.Black,
                fontSize = 16.sp,
                fontWeight = FontWeight.Normal)
            )
        },
        singleLine = false,
        maxLines = 100,
        trailingIcon = {
            Box(
                modifier = Modifier.weight(0.9f),
                contentAlignment = Alignment.TopStart
            ) {
            DisableSelection {
                    Text("find",)
                }
            }
        },
        modifier = Modifier.padding(start = 8.dp, 8.dp, 8.dp, 1.dp).fillMaxWidth(1f)
    )
  }
}

/** обработчик команд **/
fun commandExecutive(command: String) {
  logger.info{ "executive $command" }
  TODO()
}

/** указатель того, что поиск не нужен **/
fun findNope(id: MutableState<Int>, type: MutableState<FIND>) {
  type.value = FIND.NOPE
}

fun loadScheduleFromCache(file : String,scheduleType : Any) : Any? {
  logger.info { "loadScheduleFromCache" }
  val jsonString = readFile(Dir.cache,file)
  return if ( jsonString.isNotEmpty() )
     fromJson(jsonString, scheduleType)
```

```
    else {
        logger.error { "file not reading" }
        null
    }
}

/** загружает данные расписания с использованием кэша **/
fun findTeacher(name: String, ShearId: MutableState<Int>, type: MutableState<FIND>) {
    val id = Data.getTeacherId(name)
    if ( id != -1 ) {
        if ( Data.getTeacher(id) == null )
            loadTeacher(id, ShearId, type)
        else {
            type.value      = FIND.TEACHER
            ShearId.value   = id
        }
    }
    else
        logger.error { "findForGroup teacher not found" }
}

fun findClassroom(name: String) {

}

fun findClassroomWithTeacher(name: String) {

}

fun DetermineSearchType(str: String) : FIND {
    return when {
        "-".toRegex().containsMatchIn(str) -> FIND.CLASSROOM
        "(\\d){4}".toRegex().containsMatchIn(str) -> FIND.GROUP
        !"""[${'$'}M${'$'}K${'$'}КВ${'$'}КС${'$'}ВЦ${'$'}МК${'$'}КСВ^А^В^И^М\
d]""".toRegex().containsMatchIn(str) -> FIND.TEACHER
        else -> FIND.NOPE
    }
}



// @filename /view/SliderView.kt
```

```
package org.suai.schedule.view

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.padding
import androidx.compose.material.ButtonDefaults
import androidx.compose.material.TextButton
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp
import org.suai.schedule.model.Theme
import org.suai.schedule.style.Subtitle1

@Composable
fun SliderView(noActiveColor: Color, activeColor: Color, active: MutableState<Int>, options:
List<String>, modifier: Modifier = Modifier, theme: MutableState<Theme>) {
    var color : Color
    val maxSize: Float = (1.0 / options.size).toFloat()
    logger.info { "recompose SliderView [${options.size}]" }
    Row {
        for ( i in options.indices ) {
            color = if ( i == active.value )
                activeColor
            else
                noActiveColor
            TextButton(
                modifier = modifier.weight(maxSize)
                                   .padding(5.dp)
                                   .background(color),
                elevation = ButtonDefaults.elevation(),
                onClick = {
                    logger.info{ "SliderView selection $i" }
                    active.value = i
                }
            ) {
                Subtitle1(options[i], color = theme.value.font)
            }
        }
    }
}
```

```
// @filename /view/TopBarView.kt
package org.suai.schedule.view

import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.text.selection.SelectionContainer
import androidx.compose.material.Button
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import org.suai.schedule.core.data.Data.deleteLessonForGroup
import org.suai.schedule.core.settings.Settings
import org.suai.schedule.model.Link
import org.suai.schedule.model.Theme

@Composable
fun TopBarView(id: MutableState<Int>,
 type: MutableState<FIND>, link: MutableState<Link?>, recompose: MutableState<Boolean>, theme:
MutableState<Theme>) {
    if ( link.value != null ) {
        logger.info { "recompose TopBarView" }
        val action = remember { mutableStateOf(0) }
        SliderView(
            noActiveColor   = theme.value.first,
            activeColor     = theme.value.second,
            active          = action,
            options         = listOf("", "Export", "Edit", "Delete"),
            modifier        = Modifier.padding(horizontal = 6.dp),
            theme           = theme
        )
        logger.info { "TopBarView slider select [${action.value}]" }
        when (action.value) {
            1 -> {
                val string = "!import" + when (link.value!!.viewType) {
```

```
                //FIND.GROUP -> Export.groupFromString(link.value!!.viewMainId)
                //FIND.TEACHER -> Export.teacherFromString(link.value!!.viewMainId)
                else -> "Export not support"
            }
            SelectionContainer {
                Text(
                    text = string,
                    style = TextStyle(  color = Color.Black,
                        fontSize = 5.sp,
                        fontWeight = FontWeight.Medium)
                )
            }
            Button(
                onClick = { recompose.value = true }
            ) {
                Text("X")
            }
        }
        2 -> TODO()
        3 -> {
            deleteLessonForGroup(link.value!!)
            link.value = null
            Settings.linkToSelect.value = null
            recompose.value = true
        }           // Удаление выбранной группы
    }
  }
}

// @filename /view/scheduleView/DayView.kt
package org.suai.schedule.view.scheduleView

import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.paddingFromBaseline
import androidx.compose.material.Card
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import mu.KotlinLogging
```

```
import org.suai.schedule.model.Link
import org.suai.schedule.model.Theme
import org.suai.schedule.model.schedule.Day
import org.suai.schedule.model.schedule.full.fullDayModel
import org.suai.schedule.style.Subtitle4
import org.suai.schedule.view.logger

/** отрисовка дня **/
@Composable
fun DayView(dayF: fullDayModel, day: Day, link : Link, theme: MutableState<Theme>) {
   KotlinLogging.logger{}.info { "recompose DayView $day" }
   link.viewDay = day.nameId
   Card(
      elevation = 6.dp,
   ) {
      Modifier.padding(3.dp)
            .fillMaxWidth()
            .paddingFromBaseline(top = 10.dp)
      Column {
         Subtitle4(dayF.name, color = theme.value.font)
         for (i in 0 until day.pair.size)
            PairView(dayF.pair[i], day.pair[i], link.getCopy(), dayF.name, theme)
      }
   }
}


// @filename /view/scheduleView/LessonEditView.kt
package org.suai.schedule.view.scheduleView

import androidx.compose.runtime.Composable
import org.suai.schedule.model.schedule.Lesson
/*
@Composable
fun LessonEditView(lesson : Lesson) {
   val fullLesson      = lesson.toFull()

   rememberLesson.init(
      fullLesson.name,
      fullLesson.type,
      fullLesson.building,
      fullLesson.classroom,
```

```
        fullLesson.teacherName,
        fullLesson.groupNumber
    )
    //logger.info { " ${Settings.viewType} ${Settings.viewMainId} ${Settings.viewDay} $
{Settings.viewTime} ${Settings.viewTypeDay}" }
}
*/

// @filename /view/scheduleView/LessonView.kt
package org.suai.schedule.view.scheduleView

import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import org.suai.schedule.model.Theme
import org.suai.schedule.model.schedule.full.fullLessonModel
import org.suai.schedule.style.Subtitle1
import org.suai.schedule.style.Subtitle2
import org.suai.schedule.style.Subtitle3
import org.suai.schedule.style.Subtitle4

@Composable
fun LessonView(lessonF: fullLessonModel, theme: MutableState<Theme>) {
    Row {
        Column {
            Row {
                Subtitle3(lessonF.type, theme.value.font, Modifier.padding(end = 2.dp))  // ПР
                Subtitle4(lessonF.name, theme.value.font)                               // Вычислительная
математика
            }
            LazyRow(
                Modifier.fillMaxWidth()
            ) {
                for (teacher in lessonF.teacherName)
                    item { Subtitle1(teacher, theme.value.font) }                       // Ассаул В.Н
            }
```

```
            Subtitle2(lessonF.building, theme.value.sabFont)
            LazyRow {
                for ( classroom in lessonF.classroom )
                    item { Subtitle2(classroom, theme.value.sabFont, Modifier.padding(3.dp)) }
            }
            LazyRow {
                for ( group in lessonF.groupNumber )
                    item { Subtitle1(group, theme.value.font, Modifier.padding(3.dp)) }
            }
        }
    }
}

// @filename /view/scheduleView/PairView.kt
package org.suai.schedule.view.scheduleView

import androidx.compose.foundation.background
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.material.Card
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import mu.KotlinLogging
import org.suai.schedule.core.settings.Settings
import org.suai.schedule.model.Link
import org.suai.schedule.model.Theme
import org.suai.schedule.model.schedule.PAIR_TIME
import org.suai.schedule.model.schedule.full.fullPairModel
import org.suai.schedule.model.schedule.Pair
import org.suai.schedule.style.Subtitle3
import org.suai.schedule.style.Subtitle4
import org.suai.schedule.view.logger

@Composable
fun PairView(pairF: fullPairModel, pair: Pair, link: Link, dayName: String, theme:
MutableState<Theme>) {
    link.viewTime = pair.timeId
    KotlinLogging.logger{}.info { "recompose pairView $pair" }

    Row(
```

```
    Modifier.padding(2.dp).fillMaxWidth()
) {
    Column {
        if ( !dayName.contains(pairF.time, true) ) {
            Subtitle4(pairF.time, theme.value.font) // № Пары
            pairF.timeStart?.let { Subtitle3(it.toString(), theme.value.font) }
            pairF.timeEnd?.let   { Subtitle3(it.toString(), theme.value.font) }
        }
    }
    if ( pair.equally ) {                          // Одна пара
        Box( Modifier.background(theme.value.lesson)
                .clickable {
                logger.info { link }
                Settings.linkToSelect.value = link
            }
        )
            pair.high?.let { _ ->  LessonView( pairF.high!!, theme) }
    } else {
        Card(
            elevation = 3.dp
        ) {
            Column {
                Box( Modifier
                    .background(theme.value.topLesson)
                    .clickable {
                        link.viewTypeDay = PAIR_TIME.HIGH
                        logger.info { link }
                        Settings.linkToSelect.value = link
                    }
                ) {
                    pair.high?.let { _ -> LessonView( pairF.high!!, theme) }
                }
                Box( Modifier
                     .background(theme.value.lowLesson)
                     .clickable {
                        link.viewTypeDay = PAIR_TIME.LOW
                        logger.info { link }
                        Settings.linkToSelect.value = link
                    }
                ) {
                    pair.low?.let { _ -> LessonView(pairF.low!!, theme) }
                }
```

```
            }
        }
    }
}

// @filename /view/scheduleView/ScheduleBlockView.kt
package org.suai.schedule.view.scheduleView

import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import mu.KotlinLogging
import org.suai.schedule.core.data.Data
import org.suai.schedule.model.Link
import org.suai.schedule.model.Theme
import org.suai.schedule.style.Subtitle4
import org.suai.schedule.view.FIND

@Composable
fun ScheduleBlockView(id: MutableState<Int>, type : MutableState<FIND>, theme:
MutableState<Theme> ) {
    KotlinLogging.logger{}.info { "recompose ScheduleBlockView for ${type.value}" }
    Modifier.padding(2.dp).fillMaxWidth()
    val verticalScrollableState = rememberScrollState(0)
    Column(
        modifier = Modifier.padding(top = 15.dp).verticalScroll(verticalScrollableState),  // Отступ от
блока поиска
        horizontalAlignment = Alignment.CenterHorizontally,
        ) {
        when (type.value) {
            FIND.NOPE -> {
                logger.info { "recompose ScheduleView NOPE" }
```

```
            Text("Nope")
        }
        FIND.GROUP -> {
            val group = Data.getGroup(id.value)
            if (group != null) {
                logger.info { "recompose ScheduleView GROUP ${id.value} " }
                Subtitle4("Расписание для группы: " + group.number, theme.value.font)
                ScheduleView(group, Link(viewType = FIND.GROUP, viewMainId = id.value), theme)
            } else
                logger.error { "recompose ScheduleView GROUP NULL " }
        }
        FIND.CLASSROOM -> {
            logger.info { "recompose ScheduleView CLASSROOM" }
        }
        FIND.TEACHER -> {
            val teacher = Data.getTeacher(id.value)
            if (teacher != null) {
                logger.info { "recompose ScheduleView TEACHER ${id.value} " }
                Subtitle4("Расписание преподавателя: " + teacher.name, theme.value.font)
                ScheduleView(teacher, Link(viewType = FIND.TEACHER, viewMainId = id.value),
theme)
            } else
                logger.error { "recompose ScheduleView GROUP NULL " }
        }
        FIND.CROSSING_WITH_TEACHER -> {
            logger.info { "recompose ScheduleView TEACHER" }
            TODO()
        }
    }
  }
}


// @filename /view/scheduleView/ScheduleView.kt
package org.suai.schedule.view.scheduleView

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.ui.Modifier
import mu.KotlinLogging
```

```
import org.suai.schedule.model.Link
import org.suai.schedule.model.Theme
import org.suai.schedule.model.schedule.Schedule

internal val logger = KotlinLogging.logger{}

@Composable
fun ScheduleView(schedule: Schedule, link : Link, theme: MutableState<Theme> ) {
    for (i in 0 until schedule.day.size) {
        Box(
            Modifier.background(theme.value.day)
        ) {
            DayView(schedule.day[i].toFull(), schedule.day[i], link.getCopy(), theme)
        }
    }
}
```