

Вадим Салаватов

# Реализация мультиплатформенного доступа к файловым хранилищам на языке Kotlin

Выпускная квалификационная работа

Научный руководитель: **(TODO: )** В. Н. Брагилевский

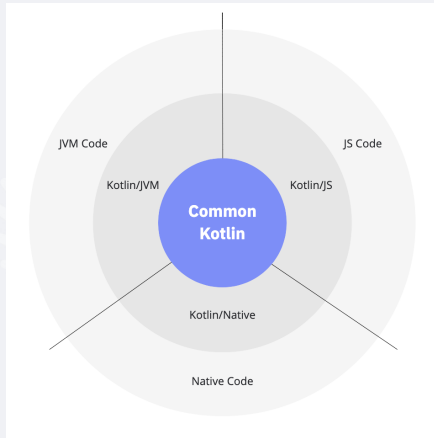
14 июня 2022



Факультет математики и компьютерных наук СПбГУ  
Программа «Современное программирование»

# Введение в предметную область

- Kotlin Multiplatform
- Файловые системы
- Хотим писать код в общем модуле
- Как работать с файловыми хранилищами в веб-браузере?



# Обзор основных аналогов

- okio
  - + мультиплатформенность
  - + активно развивается
  - + FileSystem...
  - ...но для JS требует Node.js



# Обзор основных аналогов

- okio
  - + мультиплатформенность
  - + активно развивается
  - + FileSystem...
    - ...но для JS требует Node.js
- korlibs/korio
  - + мультиплатформенность
  - + интерфейс VFS, несколько реализаций
    - последнее обновление почти год назад
    - перегруженность VFS
  - ± для JS есть VFS на основе localStorage



# Обзор основных аналогов

- okio
  - + мультиплатформенность
  - + активно развивается
  - + FileSystem...
    - ...но для JS требует Node.js
- korlibs/korio
  - + мультиплатформенность
  - + интерфейс VFS, несколько реализаций
    - последнее обновление почти год назад
    - перегруженность VFS
  - ± для JS есть VFS на основе localStorage

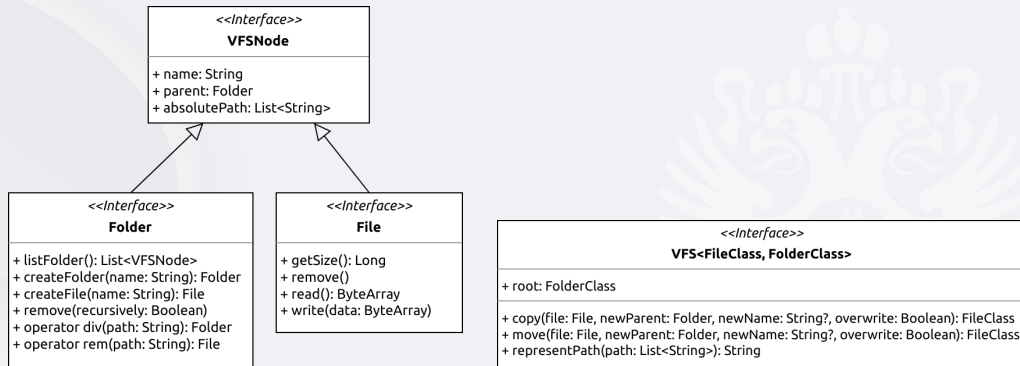


# Постановка задачи

1. Разработать мультиплатформенную библиотеку для работы с файловыми хранилищами
2. Поддержать платформы JVM, Android, JS (browser)
3. Поддержать как минимум одно облачное хранилище
4. Обеспечить простую расширяемость как в плане поддержки новых хранилищ, так и в плане предоставляемой функциональности



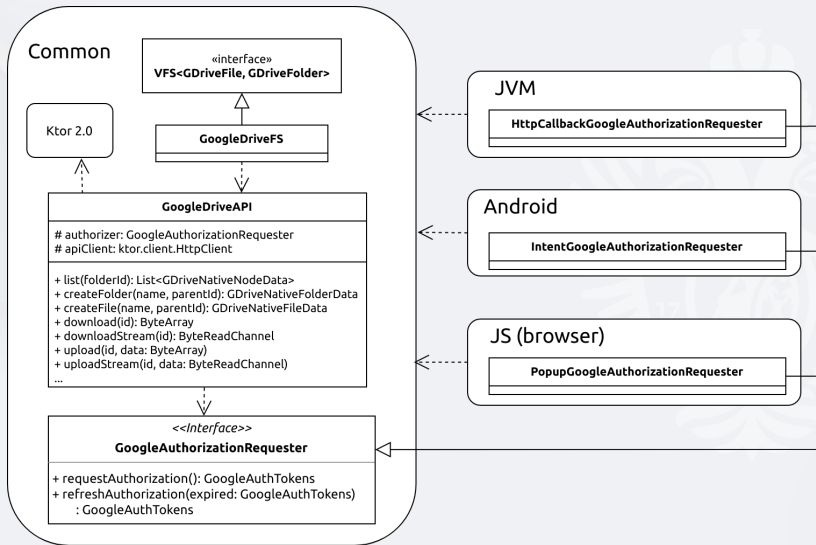
# Архитектура библиотеки



Все методы (кроме representPath) помечены suspend

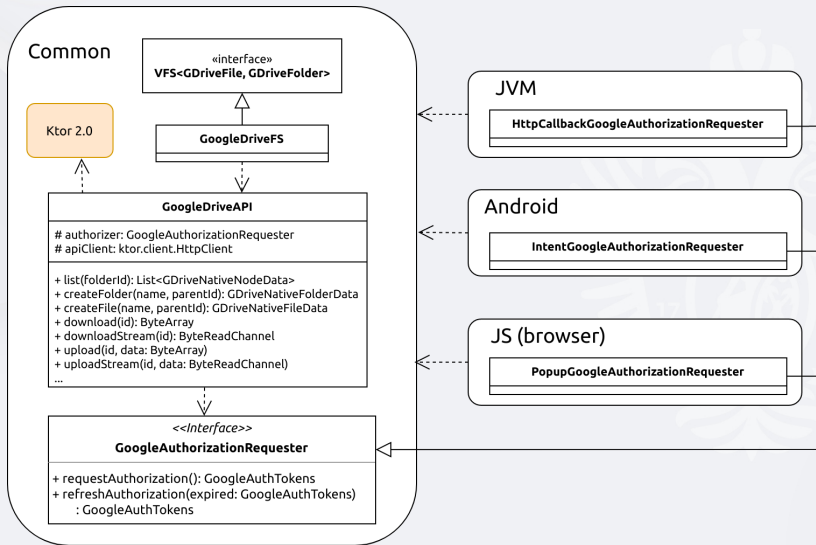


# GoogleDriveFS

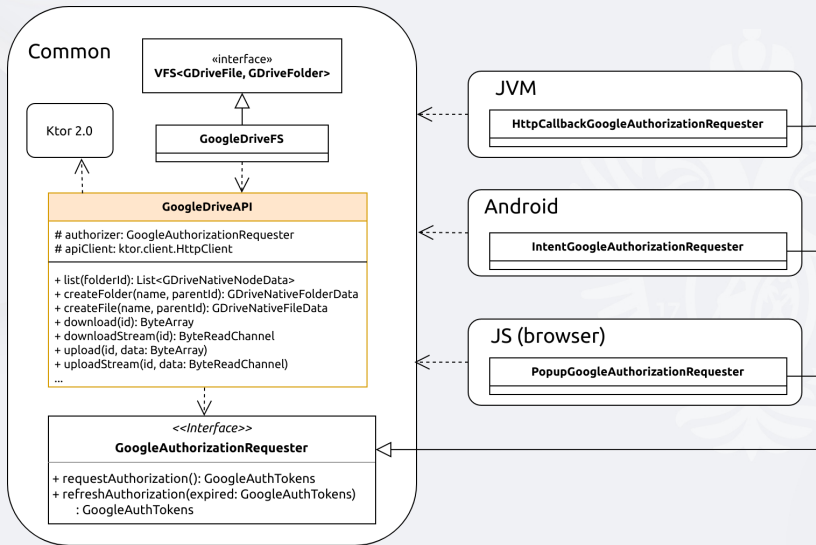




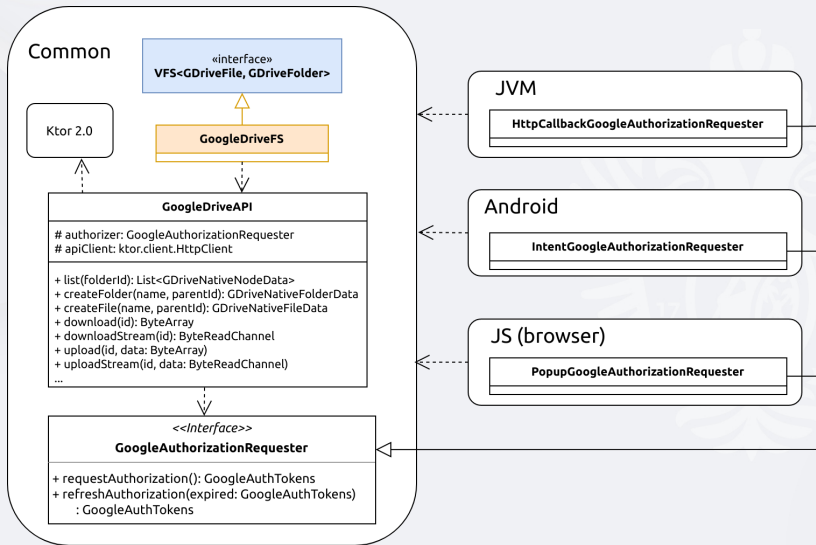
# GoogleDriveFS



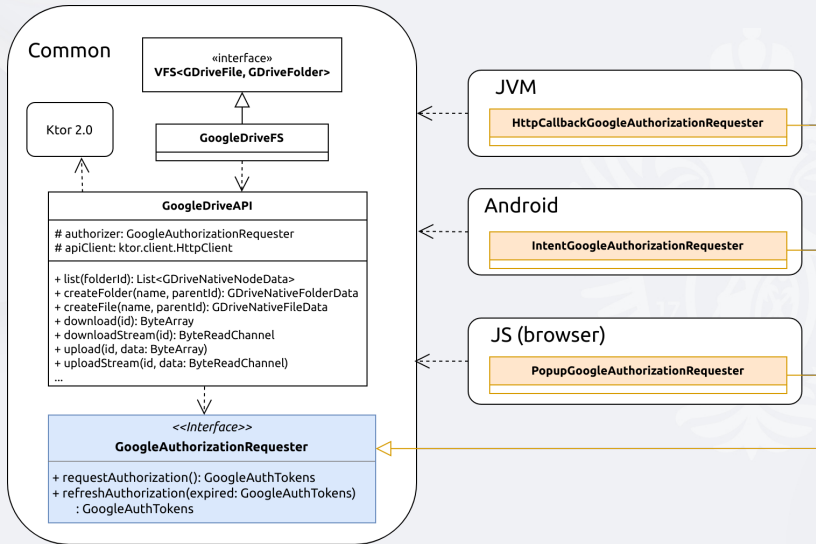
# GoogleDriveFS



# GoogleDriveFS



# GoogleDriveFS



# Расширения

<i>&lt;&lt;Interface&gt;&gt;</i> <b>StreamingIO</b>
+ readStream(): ByteReadChannel + writeStream(data: ByteReadChannel)

SystemFS и GoogleDriveFS реализуют потоковое чтение и запись



# Сводная таблица поддерживаемых возможностей

Название	Платформа			Расширения
	JVM	Android	JS (browser)	StreamingIO
SystemFS	да	да		да
GoogleDriveFS	да	да	да	да
SqliteFS		да		



# Приложения на основе библиотеки

- Multieditor<sup>1</sup> — мультиплатформенный текстовый редактор, работающий на платформах JVM, Android, Web
- gdrive-cli<sup>2</sup> — интерфейс командной строки для работы с содержимым Google Drive, позволяет скачивать и загружать файлы в потоковом режиме

---

<sup>1</sup><https://github.com/vsalavatov/multieditor>

<sup>2</sup><https://github.com/vsalavatov/gdrive-cli>



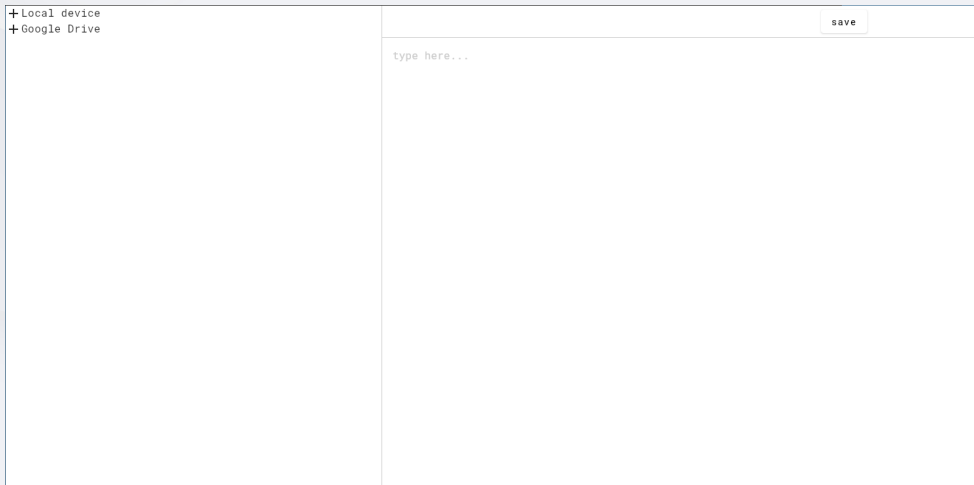
# Multieditor

- Вся логика работы с файлами в общем модуле
- Код графических интерфейсов для JVM и Android написан с помощью Compose в общем для них модуле
- Код графического интерфейса для JS (browser) написан на Compose for Web

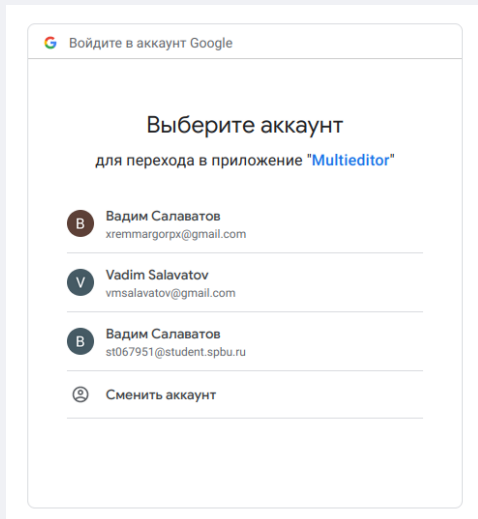




# Multieditor JVM



# Multieditor JVM



# Multieditor JVM

- Google Drive +
  - cat-video +
    - cat-video.mp4
  - sample folder +
    - code.c
    - файл.txt
    - code.cpp
    - img.xpm
    - test.txt
    - copy.kt
  - Colab Notebooks +
    - ResNet-20 Consensus Notebook
  - picture.jpg
- Local device +
  - run +
  - lib +
  - var +
  - tmp +
  - boot +
  - srv +
  - bin +
  - lib64 +
  - mnt +
  - sbin +
  - lost+found +
  - sys +
  - opt +
  - etc +
  - examples +
  - proc +
  - dev +
  - usr +
  - home +
    - vsalavator +
      - .gradle +

copy.kt

save

```
package dev.salavatov.multieditor

import dev.salavatov.multieditor.multifs.CacheGoogleAuthorizationRequester
import dev.salavatov.multifs.cloud.googledrive.GoogleAppCredentials
import dev.salavatov.multifs.cloud.googledrive.GoogleDriveAPI
import dev.salavatov.multifs.cloud.googledrive.GoogleDriveFS
import dev.salavatov.multifs.cloud.googledrive.HttpCallbackGoogleAuthorizationRequester
import dev.salavatov.multifs.systemfs.SystemFS

fun makeStorageList(): List<NamedStorageFactory> {
    val systemfs = NamedStorageFactory("Local device") { SystemFS() }
    val gdfs = NamedStorageFactory("Google Drive") {
        val googleAuth = CacheGoogleAuthorizationRequester(
            HttpCallbackGoogleAuthorizationRequester(
                GoogleAppCredentials(
                    "783177635948-ishda9322n9pk96b2uc6opp729ia0a42.apps.googleusercontent.com",
                    "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"
                ),
                GoogleDriveAPI.Companion.DriveScope.General
            )
        )
        val gapi = GoogleDriveAPI(googleAuth)

        GoogleDriveFS(gapi).also {
            it.root.listFolder() // trigger auth
        }
    }
    return listOf(systemfs, gdfs)
}
```



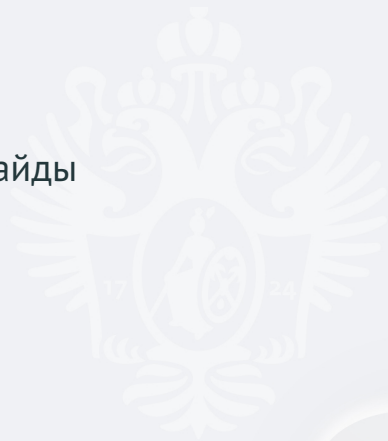
# Результаты

1. Реализована библиотека для мультиплатформенной работы с файловыми хранилищами
  2. Поддержаны три хранилища, из которых одно облачное
  3. На основе библиотеки реализовано мультиплатформенное приложение
- 

`vmsalavator@gmail.com, @vsalavator`  
`https://github.com/vsalavator/multifs`



## Дополнительные слайды



## Пример. Общий код

```
typealias Storage = VFS<out File, out Folder>
suspend fun printFiles(folder: Folder) {
    for (node in folder.listFiles()) {
        when (node) {
            is File -> println("${node.name} at ${node.absolutePath}")
            is Folder -> printFiles(node)
        }
    }
}

suspend fun printStorages(storages: List<Storage>) {
    storages.forEach { printFiles(it.root) }
}
```



## Пример. Клиентский код

```
fun getStorages(): List<Storage> {  
    val systemfs = SystemFS()  
    val googleAuth = HttpCallbackGoogleAuthorizationRequester(  
        GoogleAppCredentials(CLIENT_ID, SECRET),  
        GoogleDriveAPI.Companion.DriveScope.General  
    )  
    val gdrivefs = GoogleDriveFS(GoogleDriveAPI(googleAuth))  
    return listOf(systemfs, gdrivefs)  
}  
  
suspend fun main() {  
    printStorages(getStorages())  
}
```



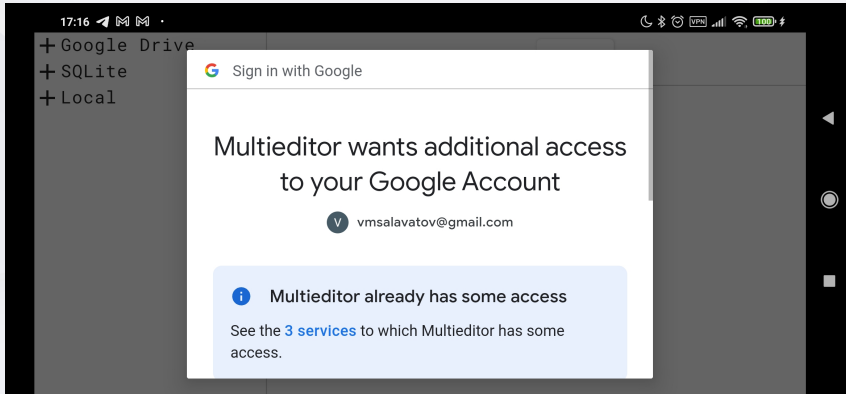
# Пример. Расширения

```
suspend fun <T> upload(fs: VFS<out T, out Folder>)  
    where T : File, T : StreamingIO {  
    val file = (fs.root / "folder" / "subfolder" % "file.txt") as T  
    val data = ByteReadChannel(  
        ByteArray(10_000_000) { it.toByteArray() }  
    )  
    file.writeStream(data)  
}  
...  
upload(gdrivefs)
```

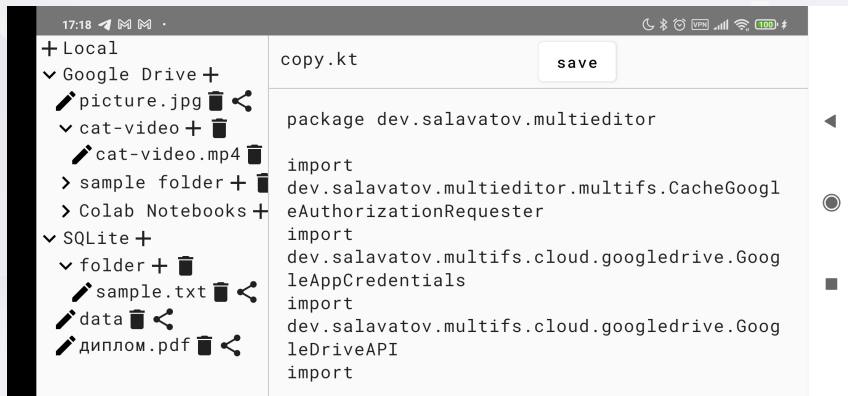




# Multieditor Android



# Multieditor Android



# Multieditor Web

← → ↺ 🏠

🔒 📄 📡 localhost:8080

▼ Google.Drive +  
🖱 picture.jpg × 🔄  
▶ cat-video  
▼ sample folder + ×  
🖱 copy.kt × 🔄  
🖱 code.c × 🔄  
🖱 файл.txt × 🔄  
🖱 code.cpp × 🔄  
🖱 img.xpm × 🔄  
🖱 test.txt × 🔄  
▼ Colab Notebooks + ×  
🖱 ResNet-20 Consensus Notebook × 🔄

copy.kt save

```
package dev.salavatov.multieditor

import dev.salavatov.multieditor.multifs.CacheGoogleAuthorizationRequester
import dev.salavatov.multifs.cloud.googledrive.GoogleAppCredentials
import dev.salavatov.multifs.cloud.googledrive.GoogleDriveAPI
import dev.salavatov.multifs.cloud.googledrive.GoogleDriveFS
import dev.salavatov.multifs.cloud.googledrive.HttpCallbackGoogleAuthorizationRequester
import dev.salavatov.multifs.systemfs.SystemFS

fun makeStorageList(): List<NamedStorageFactory> {
    val systemfs = NamedStorageFactory("Local device") { SystemFS() }
    val gdfs = NamedStorageFactory("Google Drive") {
        val googleAuth = CacheGoogleAuthorizationRequester(
            HttpCallbackGoogleAuthorizationRequester(
                GoogleAppCredentials(
                    "783177635948-ishda9322n9pk96b2uc6opp729ia0a42.apps.googleusercontent.com",
                    "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"
                ),
                GoogleDriveAPI.Companion.DriveScope.General
            )
        )
        val gapi = GoogleDriveAPI(googleAuth)

        GoogleDriveFS(gapi).also {
            it.root.listFolder() // trigger auth
        }
    }
    return listOf(systemfs, gdfs)
}
```

