Universidad de Guanajuato
Cómputo en la Nube
Proyecto Final
Hugo Ivan Lozoyo Belman , Dara Eugenia Gama Sandoval , Damián Sevilla Gallardo , Miguel Angel Vázquez Ulloa

Proyecto Final

Desarrollo de una aplicación para Android para subir , borrar y descargar archivos

Indice

Historias de Usuario

- Se necesita una manejo de cuentas de usuario
- Cada usuario debe tener acceso para borrar archivos , subir ,editar
- Otro usuario no puede borrar los archivos que no le pertenecen
- La aplicación debe correr en dispositivos android de version 5 o posterior
- Se debe utilizar un api construida por el equipo de backend

Antcendentes

- Se debe consumir apis
- Se debe diseñar los patrones
- Se contactó a los desarrolladores de la api para obtener la documentación necesaria
- Se procede a calcular los requerimientos de la aplicación

Herramientas de Desarrollo

- Android Studio
- SDK 29 / version minima 19
- Emulador AVD con Android 10
- Illustrator

Librerias

- Retrofit
- Moshi

Planificación

Con los requerimientos establecidos se opta por utilizar kotlin por su facilidad de uso con las APIS, se utilizará la librería moshi para poder recibir respuestas en formato JSON.También se acompañará el uso de la librería retrofit la cual nos permitirá transformarmar las peticiones HTTP en interfaces de kotlin.

Las herramientas a utilizar será Android studio con su SDK edición 29 y un emulador AVD con android 10.

Para la el diseño de los banners se utilizara illustrator.

Funcionamiento

Nuestra aplicación se conecta a un API , por lo cual para su funcionamiento deberá estar conectada a internet.

Las funciones principales son:

- Permite identificarse como usuario para acceder a los archivos
- En caso de no tener cuenta se dara la opcion de crear una cuenta
- El usuario identificado puede subir archivos
- El usuario identificado puede editar sus archivos
- El usuario identificado puede eliminar sus archivos

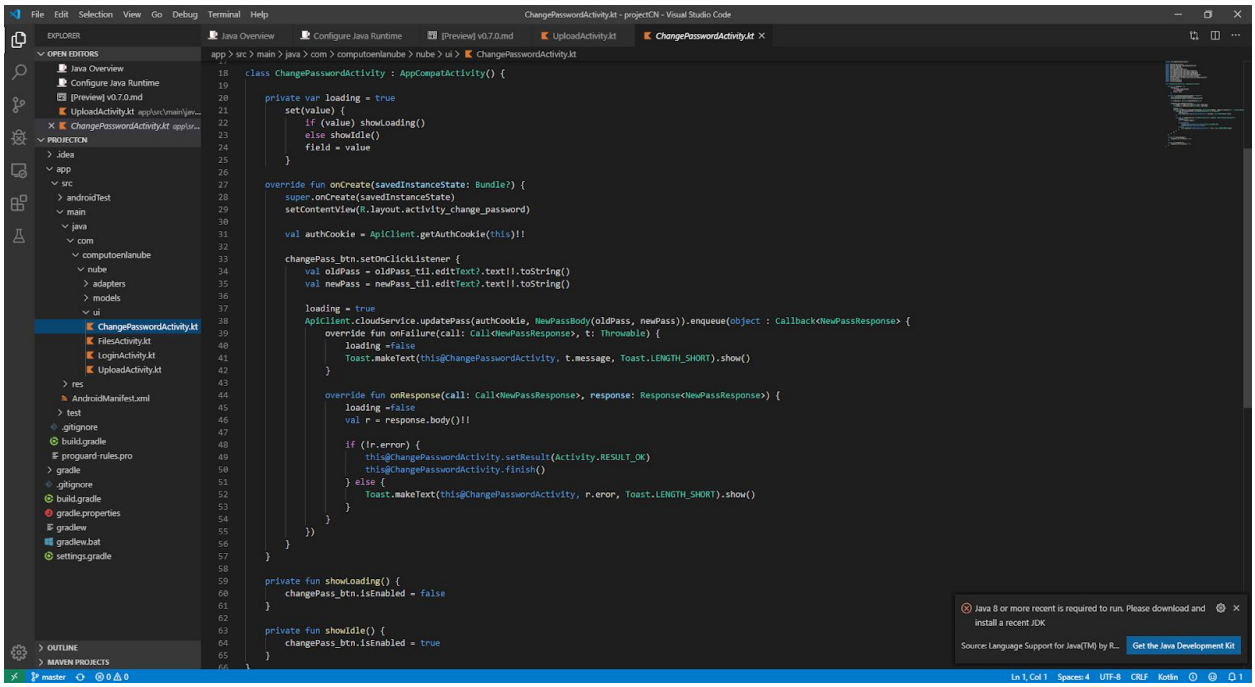
Restricciones:

- Los archivos deben ser de máximo 10 MB.
- No se pueden consumir los datos del servidor si no se está logueado.
- Sólo se pueden subir archivo, los directorios no están permitidos

Funcionamiento

La aplicación consta de 4 actividades :

- ChangePasswordActivity:
    - Esta Actividad se encarga de recuperar las contraseñas en caso de que un usuario se olvide de su contraseña
    -



- FilesActivity:
    - Se encarga de mostrar los archivos , hace las peticiones para obtener la información de la api y mostrarlas de una manera amigable

```
class FilesActivity : AppCompatActivity() {

    private lateinit var files: List<MetadataFile>
    private lateinit var authCookie: String
    private var loading = true
        set(value) {
            if (value) showLoading()
            else showIdle()
            field = value
        }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_files)

        authCookie = ApiClient.getAuthCookie(this)!!

        upload_fab.setOnClickListener {
            val intent = Intent(this, UploadActivity::class.java)
            startActivityForResult(intent, 1)
        }

        files_lv.setOnItemClickListener { adapterView, view, i, l ->
            val file = files[i]

            val dialog = AlertDialog.Builder(this).apply {
                setTitle("Archivo con extensión desconocida")
                setMessage("¿Desea descargar el archivo ${file.nom_or}")
                setPositiveButton("Si") { _, _ ->
                    downloadFile(file)
                }
                setNegativeButton("No", null)
            }

            dialog.show()
        }

        files_lv.setOnItemLongClickListener { adapterView, view, i, l ->
            val file = files[i]

            val dialog = AlertDialog.Builder(this).apply {
                setTitle("¿Desea eliminar este archivo?")
                setMessage("Se eliminará ${file.nom_or} y no se podrá recuperar")
                setPositiveButton("Si") { _: DialogInterface, _: Int ->
                    loading = true

                    ApiClient.cloudService.deleteFile(authCookie, file.id_archivos).enqueue(object : Callback<DeleteResponse> {
                        override fun onFailure(call: Call<DeleteResponse>, t: Throwable) {
```

```
private fun logout() {
    ApiClient.cloudService.logout(authCookie)
    ApiClient.deleteAuthCookie(this)

    val intent = Intent(this, LoginActivity::class.java)
    startActivity(intent)
    this.finish()
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    when {
        requestCode == 1 && resultCode == Activity.RESULT_OK -> {
            getFilesAndShowInListView()
        }

        requestCode == 2 && resultCode == Activity.RESULT_OK -> {
            logout()
        }
    }

    super.onActivityResult(requestCode, resultCode, data)
}

private fun getFilesAndShowInListView() {
    ApiClient.cloudService.getAllFiles(authCookie).enqueue(object : Callback<List<MetadataFile>> {
        override fun onFailure(call: Call<List<MetadataFile>>, t: Throwable) {
            AlertDialog.Builder(this@FilesActivity).apply {
                setTitle("Error al obtener los archivos")
                setMessage("Ocurrio un error al obtener los archivos: ${t.message}")
                setPositiveButton("Cerrar sesión") { _,_ ->
                    logout()
                }
                setCancelable(false)
            }.show()

            loading = false
        }

        override fun onResponse(call: Call<List<MetadataFile>>, response: Response<List<MetadataFile>>) {
            files = response.body()!!

            loading = false
            this@FilesActivity.files_lv.adapter = FilesAdapter(this@FilesActivity, files)
        }

    })
```

- LoginActivity:
    - Se encarga de verificar las credenciales de los usuarios , asi como de permitirse registrar a los nuevos usuarios

```
class LoginActivity : AppCompatActivity() {

    private var loading = false
    set(value) {
        if (value) showLoading()
        else showIdle()
        field = value
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        login_btn.setOnClickListener {
            val user = User(userName_til.editText!!.text.toString(), pass_til.editText!!.text.toString())

            loading = true
            login(user)
        }

        signup_btn.setOnClickListener {
            val user = User(userName_til.editText!!.text.toString(), pass_til.editText!!.text.toString())

            loading = true
            ApiClient.cloudService.signUp(user).enqueue(object : Callback<AddResponse> {
                override fun onFailure(call: Call<AddResponse>, t: Throwable) {
                    loading = false

                    Toast.makeText(this@LoginActivity, t.message, Toast.LENGTH_SHORT).show()
                }

                override fun onResponse(call: Call<AddResponse>, response: Response<AddResponse>) {
                    loading = false

                    val body = response.body()

                    when {
                        body?.add == null && body?.status == null -> ApiClient.showError(this@LoginActivity, response.code())

                        body.add == null || !body.add -> Toast.makeText(this@LoginActivity, body.status, Toast.LENGTH_SHORT).show()

                        body.add -> login(user)

                    }
                }
```

```
private fun showLoading() {
    userName_til.editText!!.isEnabled = false
    pass_til.editText!!.isEnabled = false
    login_btn.visibility = View.INVISIBLE
    signup_btn.visibility = View.INVISIBLE
    progressBar.visibility = View.VISIBLE
}

private fun showIdle() {
    userName_til.editText!!.isEnabled = true
    pass_til.editText!!.isEnabled = true
    login_btn.visibility = View.VISIBLE
    signup_btn.visibility = View.VISIBLE
    progressBar.visibility = View.INVISIBLE
}

private fun startApp() {
    val intent = Intent(this@LoginActivity, FilesActivity::class.java)
    startActivity(intent)
    this@LoginActivity.finish()
}

private fun login(user: User) {
    ApiClient.cloudService.login(user).enqueue(object : Callback<LogResponse> {
        override fun onFailure(call: Call<LogResponse>, t: Throwable) {
            loading = false
            Toast.makeText(this@LoginActivity, t.message, Toast.LENGTH_LONG).show()
        }

        override fun onResponse(call: Call<LogResponse>, response: Response<LogResponse>) {
            loading = false

            val body = response.body()

            when {
                body?.log == null -> ApiClient.showError(this@LoginActivity,  response.code())

                body.log -> {
                    ApiClient.setAuthCookie(this@LoginActivity, response.headers().get("Set-Cookie")!!)
                    startApp()
                }

                body.log == false -> Toast.makeText(this@LoginActivity, body.status, Toast.LENGTH_SHORT).show()
            }
        }

    })
}
```

- UploadActivity
    - Se encarga de subir los archivos ,verifica el tamaño y les asigna un dueño

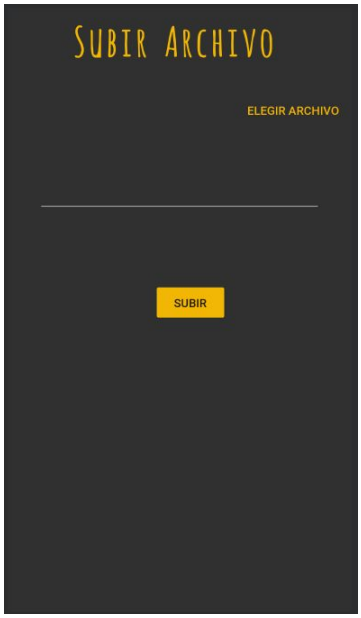
```
class UploadActivity : AppCompatActivity() {

    private val PICKFILE_REQUEST_CODE = 100
    private var uri: Uri? = null
    private var fileName: String? = null
    private var loading = false
        set(value) {
            if (value) showLoading()
            else showIdle()
            field = value
        }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_upload)

        val authCookie = ApiClient.getAuthCookie(this)!!

        selectFile_btn.setOnClickListener {
            val intent = Intent(Intent.ACTION_OPEN_DOCUMENT)
            intent.type = "*/*"
            startActivityForResult(intent, PICKFILE_REQUEST_CODE)
        }

        upload_btn.setOnClickListener {
            if (uri != null) {
                val bytes = contentResolver.openInputStream(uri!!)!!.readBytes()

                if (bytes.size > 1e+7) {
                    Snackbar.make(it, "Solo se puede subir archivos menores a 10 MB", Snackbar.LENGTH_SHORT).show()
                    return@setOnClickListener
                }

                val requestBody = RequestBody.create(MediaType.parse(contentResolver.getType(uri!!)!!), bytes)
                val formDataFile = MultipartBody.Part.createFormData("file", fileName, requestBody)

                loading = true
                ApiClient.cloudService.uploadFile(authCookie, formDataFile).enqueue(object : Callback<UploadResponse>{
                    override fun onFailure(call: Call<UploadResponse>, t: Throwable) {
                        Toast.makeText(this@UploadActivity, t.message, Toast.LENGTH_SHORT).show()
                        loading = false
                    }

                    override fun onResponse(call: Call<UploadResponse>, response: Response<UploadResponse>) {
                        loading = false
                        setResult(Activity.RESULT_OK)
                        this@UploadActivity.finish()
```

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    when {
        requestCode == PICKFILE_REQUEST_CODE && resultCode == Activity.RESULT_OK -> {
            uri = data!!.data

            val cursor: Cursor? = contentResolver.query(
                uri!!, null, null, null, null, null
            )

            cursor?.use {
                // moveToFirst() returns false if the cursor has 0 rows. Very handy for
                // "if there's anything to look at, look at it" conditionals.
                if (it.moveToFirst()) {

                    // Note it's called "Display Name". This is
                    // provider-specific, and might not necessarily be the file name.
                    fileName = it.getString(it.getColumnIndex(OpenableColumns.DISPLAY_NAME))
                }

                fileName_et.setText(fileName)
            }
        }
    }

    return super.onActivityResult(requestCode, resultCode, data)
}

private fun showLoading() {
    progressBar.visibility = View.VISIBLE
    selectFile_btn.isEnabled = false
    upload_btn.isEnabled = false
}

private fun showIdle() {
    progressBar.visibility = View.INVISIBLE
    selectFile_btn.isEnabled = true
    upload_btn.isEnabled = true
}
}
```

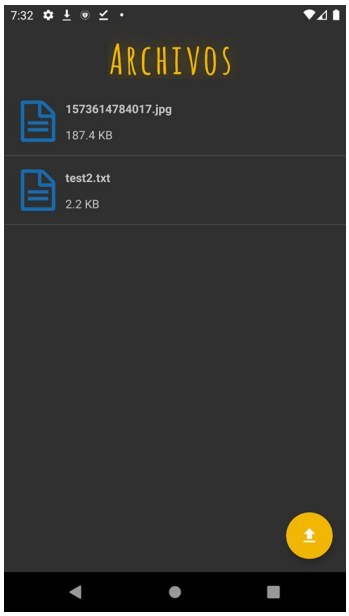
Pruebas

Pantalla de login

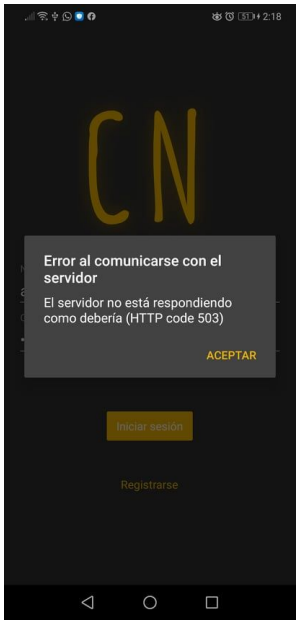
Pantalla de inicio de sesión



Pantalla de inicio , aquí se suben los archivos al presionar el botón elegir archivo



En caso de error se muestra lo siguiente:



Conclusiones

La aplicación resultó complicada de desarrollar ya que los servidores de prueba constantemente estaban en mantenimiento y la documentación era confusa , afortunadamente gracias a la comunicación entre equipos estos problemas se resolvieron y permitieron desarrollar una aplicación a la par que el back end iba creciendo.

Además nos permitió trabajar como desarrolladores ya que nos subdividimos en equipos y completamos pequeñas tareas . permitiéndonos crear una aplicación funcional y con una interfaz amigable y fácil de usar.