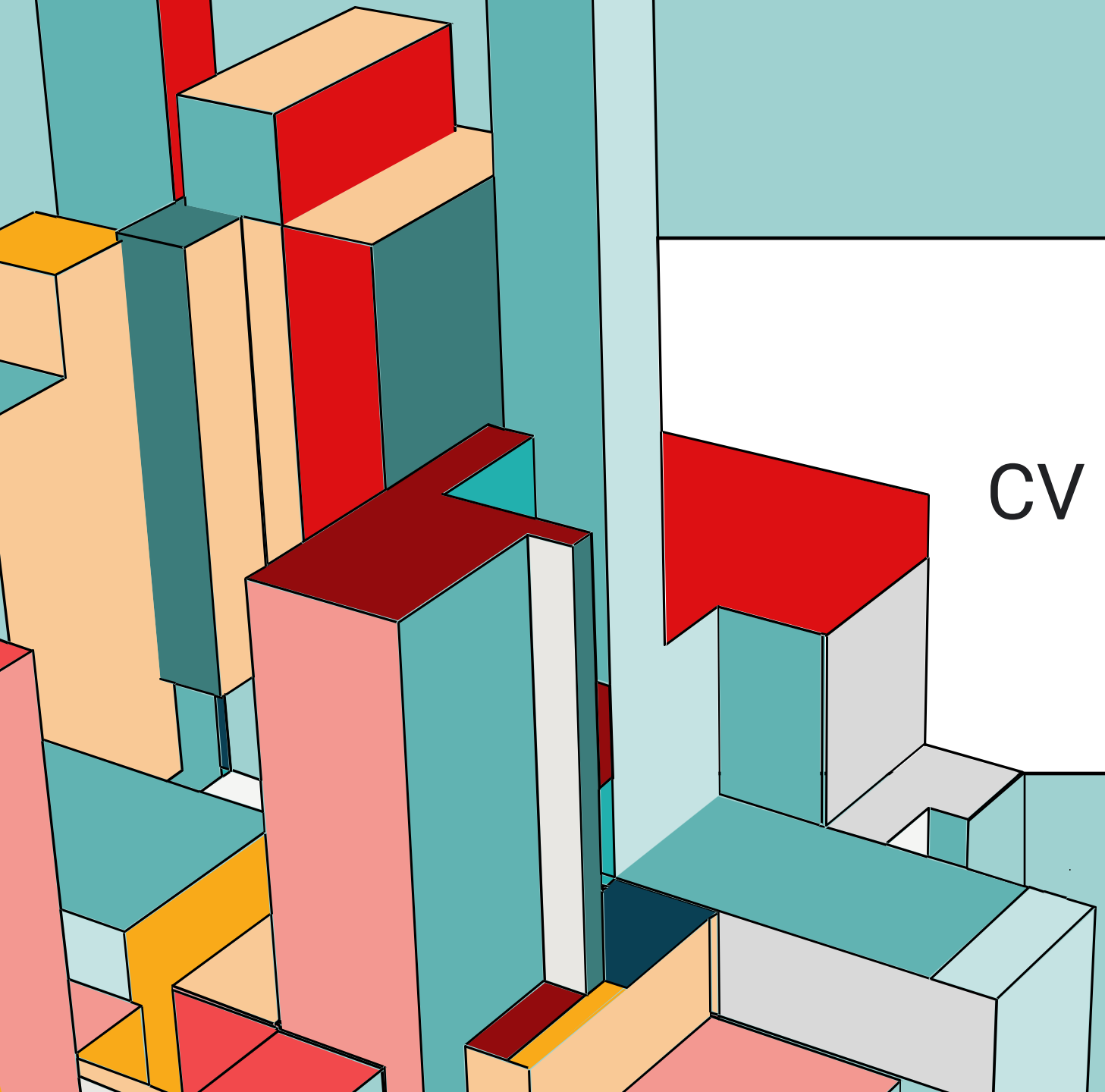


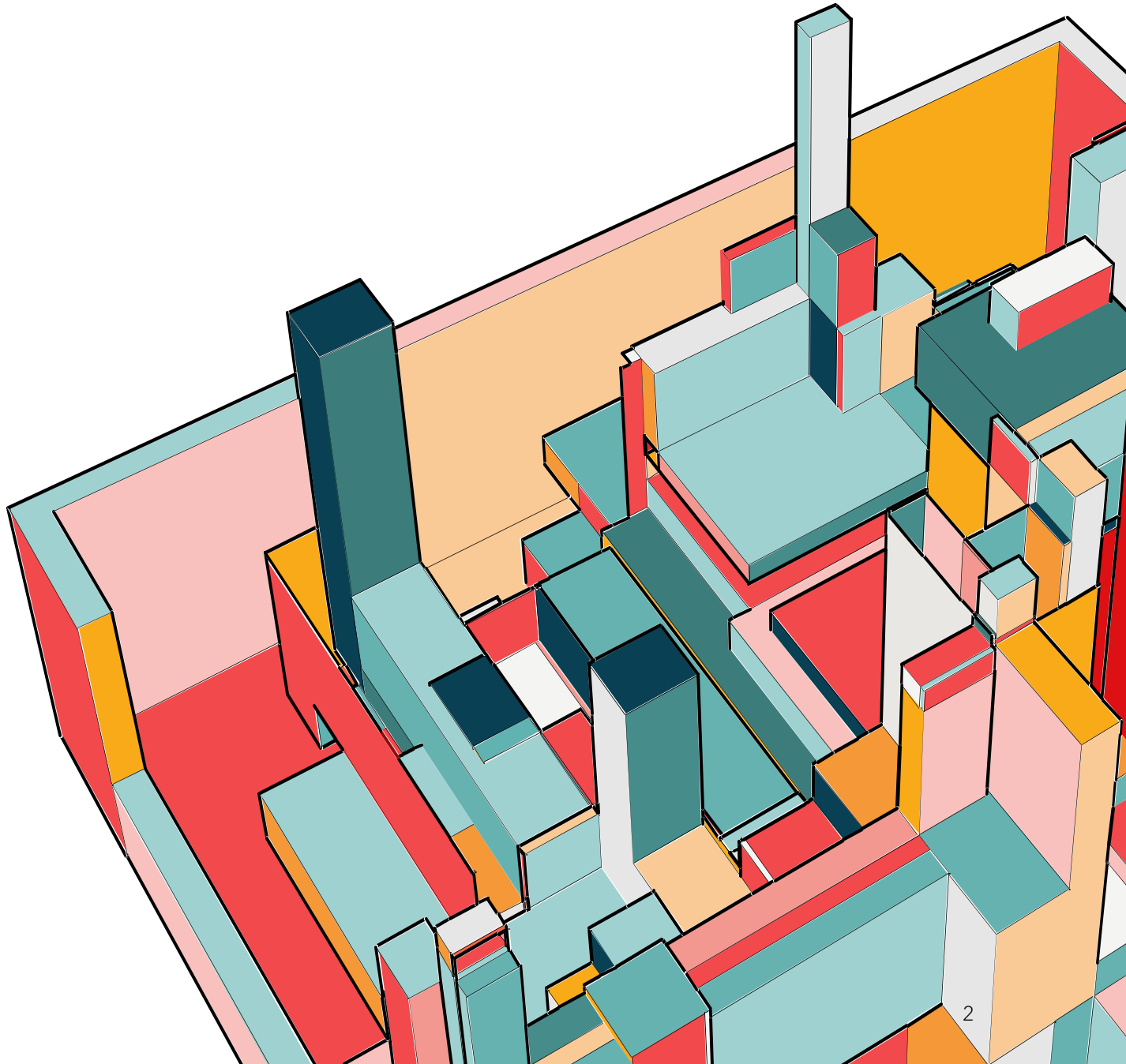
CV BUILDER APP

ABOUT US

Oscar Javier Gordillo Roncancio

Crystal McCay

Charles Grant Carpenter IV



AGENDA

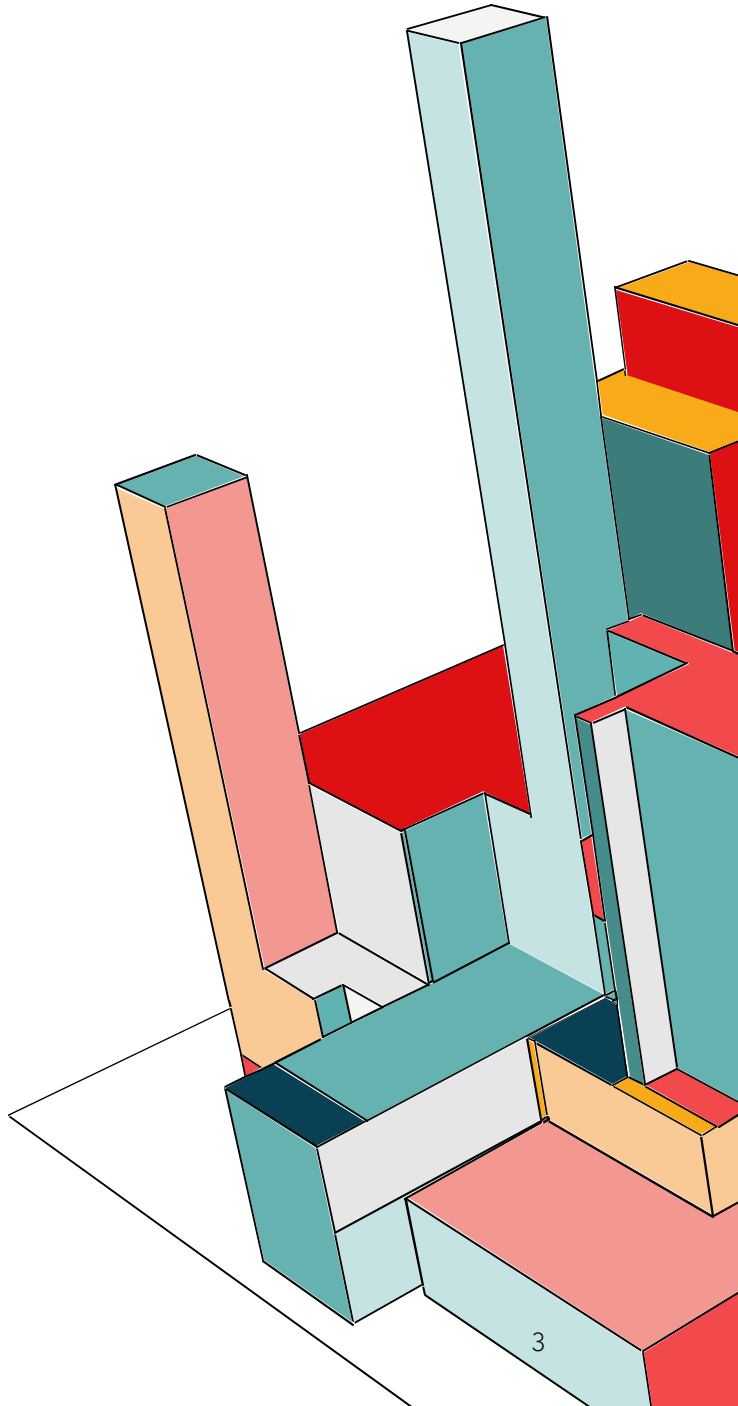
(1) App Structure

(2) Features & Modules

(3) Major Components

(4) Dependencies Overview

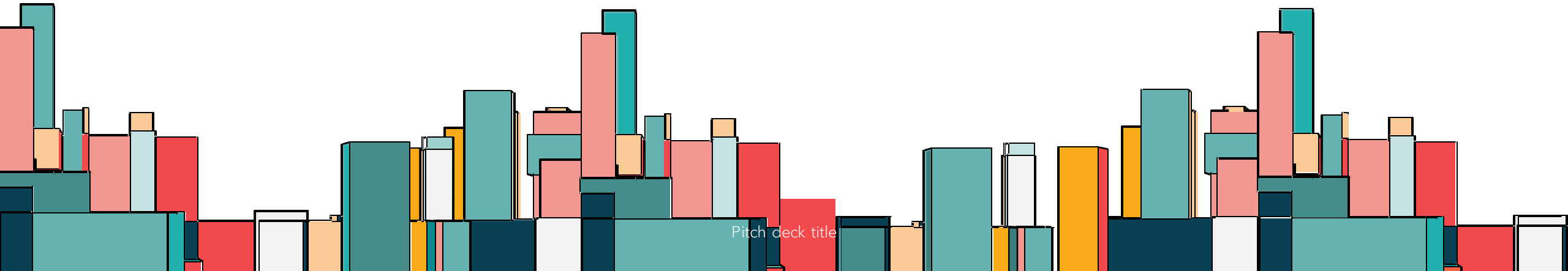
(5) Live Demo

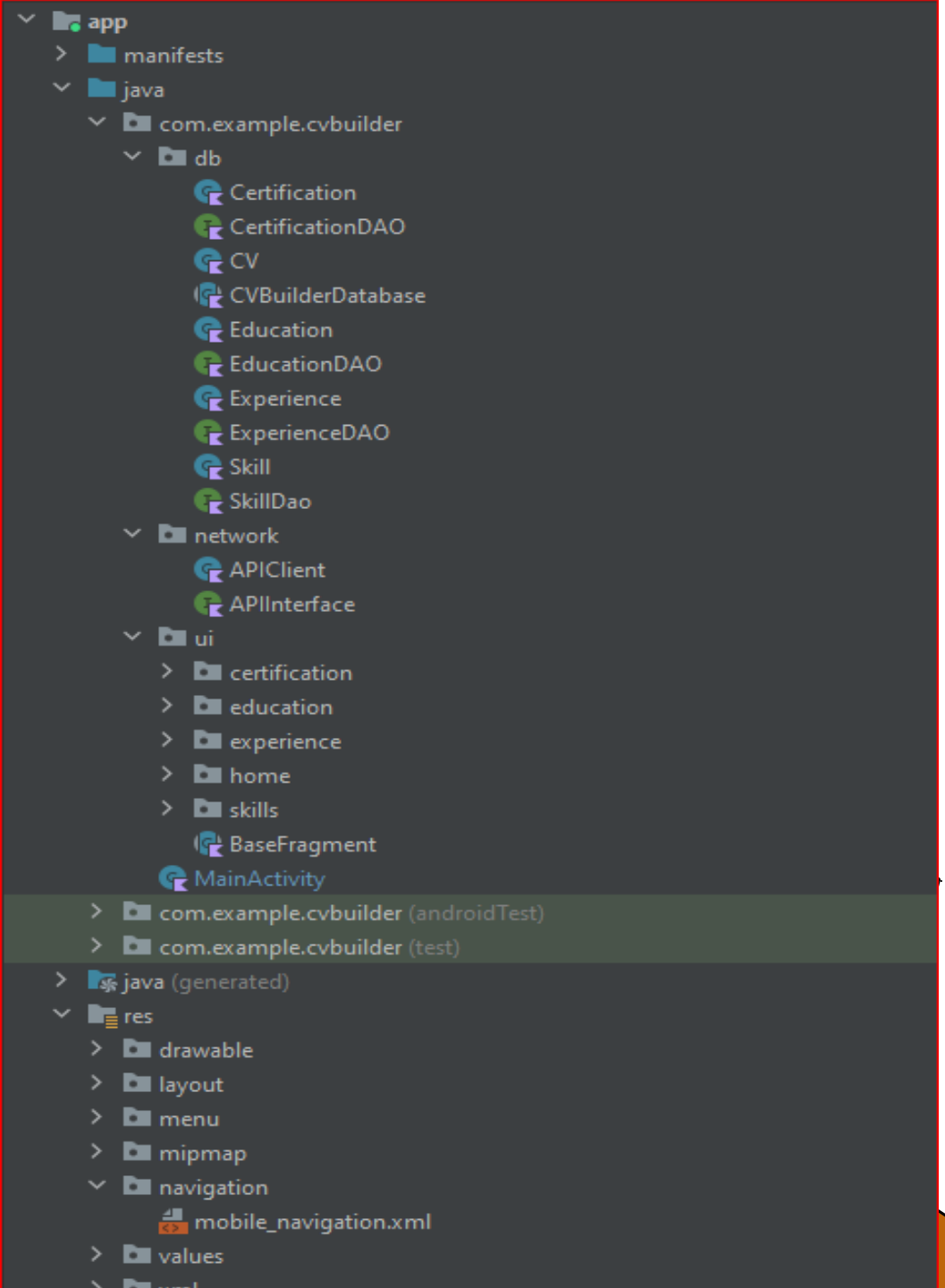


OBJECTIVE AND AUDIENCE

To have a mobile app to easily create CV on the cell phone. It allows to export to PDF using different templates, take profile picture directly from the cell phone and to backup data to the cloud.

The audience of this app are students, professionals or any person who is looking for a job and needs to create an outstanding CV





PROJECT STRUCTURE

Database Package

4 Data Classes with their respective DAO interfaces

Network Package

API call to Oracle Cloud

User Interface Package

Fragments/Adapters for each data Class used in RecylerView

Navigation Resource

Navigation Graph defining the routes between Fragments

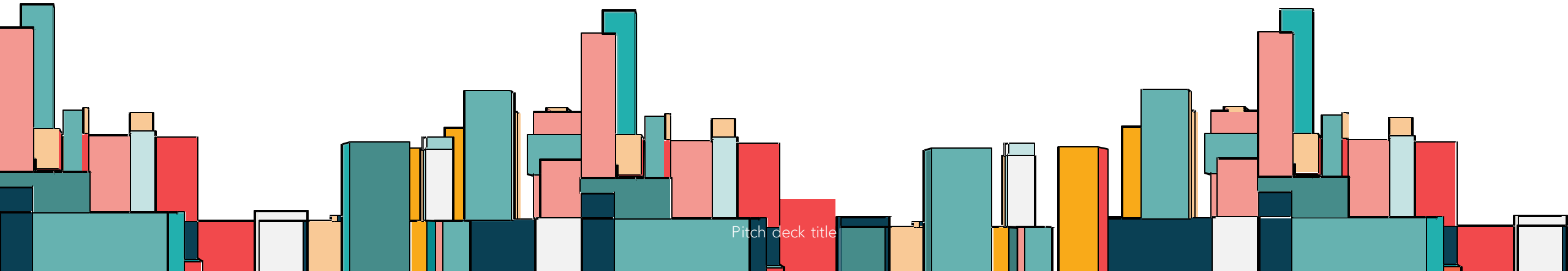
FEATURES

Database Support
-implementation of Room DB
-Implementing Kotlin Coroutines

PDF Export
-REST webservice to send JSON data, using GSON/RetroFit

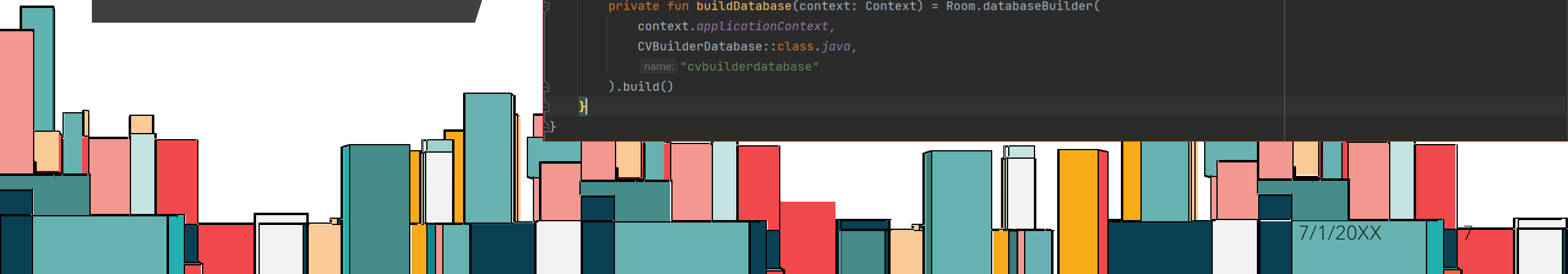
Implicit Intent
-Activity Result Launcher
-IMAGE_CAPTURE

Navigation
-Coordinate Fragment routes



ROOM DB

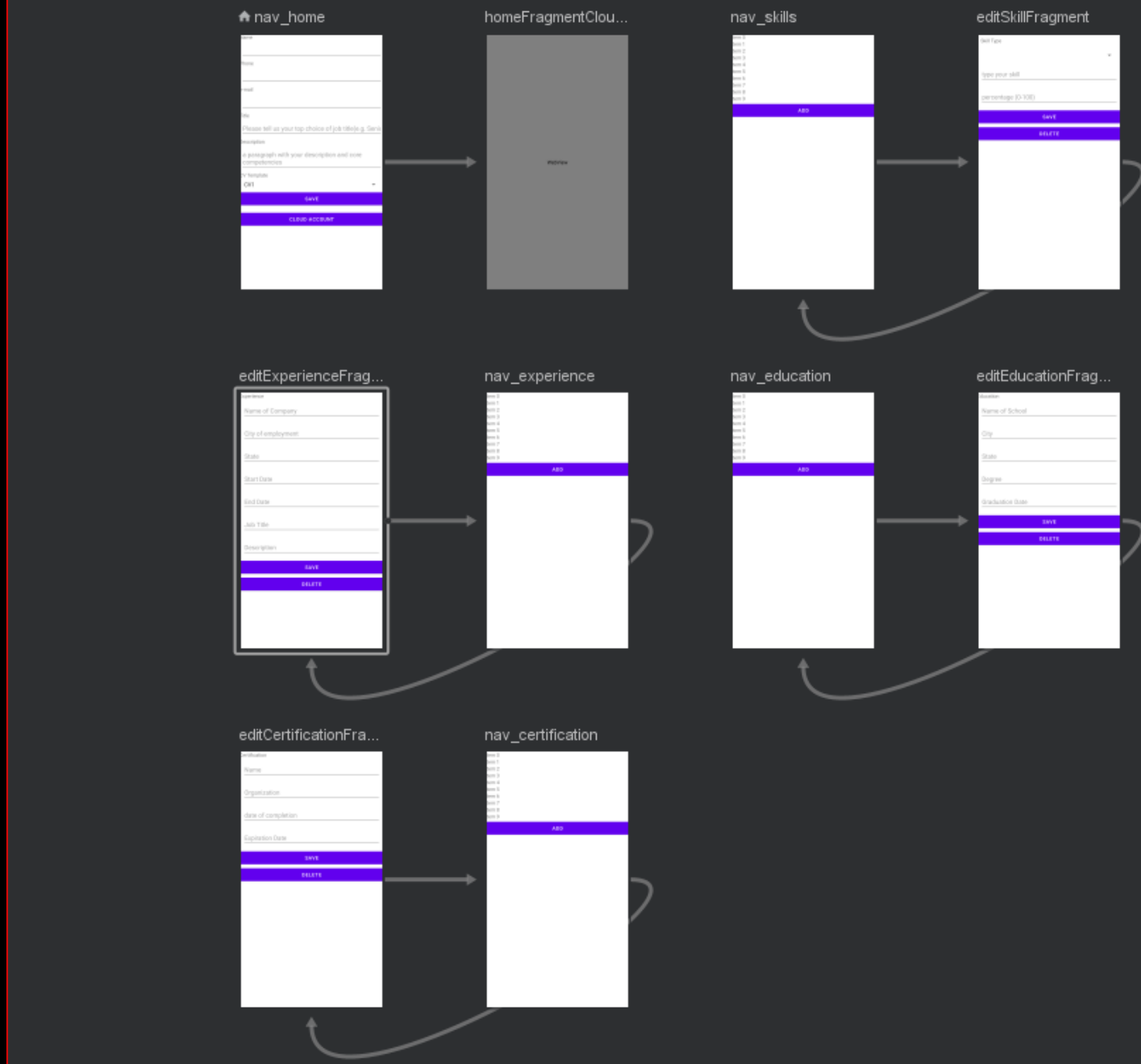
```kotlin
@Database(
    entities = [Skill::class,Certification::class,Education::class,Experience::class],
    version = 1
)
abstract class CVBuilderDatabase():RoomDatabase() { // Must Inherit from RoomDatabase

    abstract fun getSkillDao() : SkillDao // Need this function to get the Dao for the Entity
    abstract fun getExperienceDao() : ExperienceDAO
    abstract fun getEducationDao() : EducationDAO
    abstract fun getCertificationDao() : CertificationDAO
    // Build RoomDB
    companion object {
        //  means that this field is immediately made visible to other threads
        @Volatile private var instance : CVBuilderDatabase? = null
        private val LOCK = Any() // The root of the Kotlin class hierarchy. Every Kotlin class has [Any] as a superclass.
        /*  Help of ?: elvis operator check if the instance is not null return the instance,
            if it is null then synchronized block will  work, inside this also check null or not and call the function buildDatabase*/
            operator fun invoke(context: Context) = instance ?: synchronized(LOCK){
           // Create a instance also return the instance
            instance ?: buildDatabase(context).also { it: CVBuilderDatabase
                    instance = it
            }
        }


        // Function to build database
        private fun buildDatabase(context: Context) = Room.databaseBuilder(
            context.applicationContext,
            CVBuilderDatabase::class.java,
            name: "cvbuilderdatabase"
        ).build()
    }
}
```



PDF SUPPORT

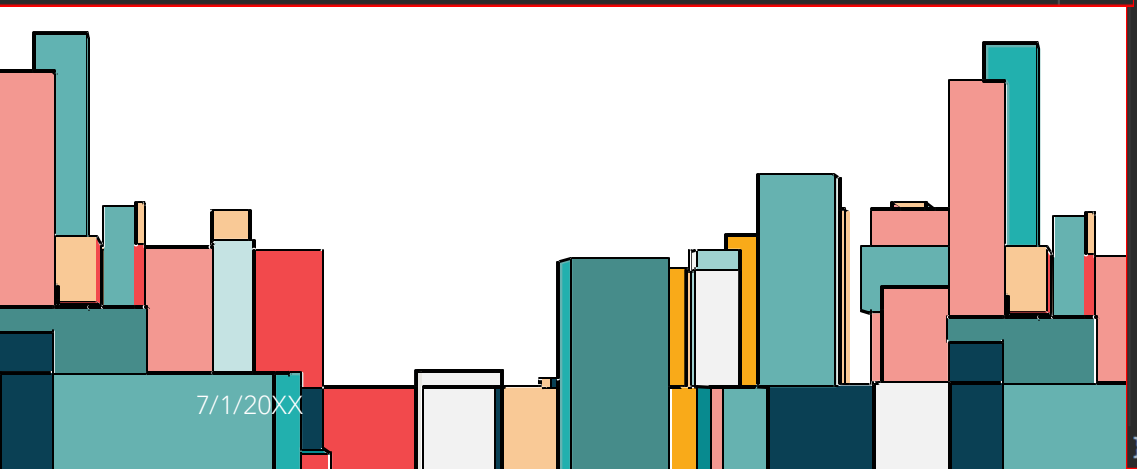
```
call.enqueue(object : Callback<String> { // Make a call to hit the server
    // hit if you receive the response--> which is to retrieve the List of ImageData
    override fun onResponse(call: Call<String?>?, response: Response<String?>?) {
        if( response!!.isSuccessful){ // Check using non null !! operator
            // The deserialized response body of a successful response ie List<Animals>
            Toast.makeText(applicationContext, text: "Downloading PDF...", Toast.LENGTH_LONG).show()
            Log.i( tag: "TEST",response.body()!!.toString())

            val spf= getSharedPreferences( name: "myspf", Context.MODE_PRIVATE)
            val spe=spf.edit()
            spe.putString("cloud_id",response.body()!!.toString())
            spe.apply()
            val url = "https://ged21c9a6c3b8b0-free1.adb.us-phoenix-1.oraclecloudapps.com/ords/r/mdp/cv-builder-app/test?p5_id="+response.body()!!.toString()
            val i = Intent(Intent.ACTION_VIEW,Uri.parse(url))
            startActivity(i)
        }
    }
    // Unable to get a response
    override fun onFailure(call: Call<String?>?, t: Throwable?) {
        // The localized description of this throwable, like getErrorMessage from throwable
        Toast.makeText(applicationContext, text: "An Error Occured  ${t?.localizedMessage}", Toast.LENGTH_LONG).show()

    }

})
```

NAVIGATION


nav_home
homeFragmentClou...
nav_skills
editSkillFragment
editExperienceFrag...
nav_experience
nav_education
editEducationFrag...
editCertificationFra...
nav_certification

IMPLICIT (CAMERA) INTENT

```
OptionsItemSelected(item: MenuItem):Boolean {
emId==R.id.action_exportPDFImage){
 Intent()
vity Action for the intent : Pick an item from the data, returning what was selected.
n = MediaStore.ACTION_IMAGE_CAPTURE
e = "image/*"
rResultCamera.launch(i)
```



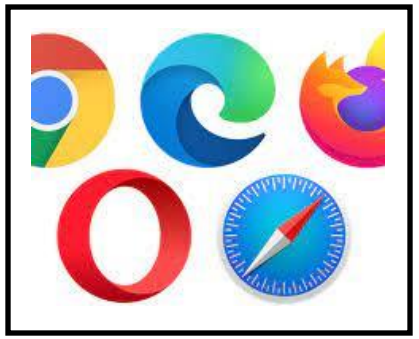

```
startforResultCamera = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
    if(it!=null) {
        //Toast.makeText(this, it.data?.data.toString(), Toast.LENGTH_LONG).show()
        //Log.i("TEST", it.data?.data.toString())
        //val uri = it.data?.data

        try {
            //val bitmap = MediaStore.Images.Media.getBitmap(contentResolver, uri)
            var bitmap = it.data?.extras?.get("data") as Bitmap
            val stream = ByteArrayOutputStream()
            bitmap.compress(Bitmap.CompressFormat.JPEG, quality: 50,stream);
            val bytes = stream.toByteArray()
            sImage= Base64.encodeToString(bytes, Base64.DEFAULT);
            Log.i( tag: "TEST",sImage)

            bitmap = getCircularBitmap(bitmap)
            binding.navView.profilePicture.setImageBitmap(bitmap)

        } catch (e: Exception) {
            // TODO: handle exception
            e.printStackTrace()
            Log.d( tag: "error",  msg: "onActivityResult: $e")
        }


    }
    else
        Toast.makeText( context: this, text: "Fail to retrieve", Toast.LENGTH_LONG).show()
}
```

GSON/RETROFIT

```
scope.launch {  this: CoroutineScope
    val spf= getSharedPreferences( name: "myspf",Context.MODE_PRIVATE)
    val skills = CVBuilderDatabase(applicationContext).getSkillDao().getAllSkills()
    val educations = CVBuilderDatabase(applicationContext).getEducationDao().getAllEducation()
    val experiences=CVBuilderDatabase(applicationContext).getExperienceDao().getAllExperience()
    val certifications=CVBuilderDatabase(applicationContext).getCertificationDao().getAllCertification()

    Log.i( tag: "TEST1",sImage)
    val cv=CV(spf.getString("name","")!!
        ,spf.getString("phone","")!!
        ,spf.getString("email","")!!
        ,spf.getString("title","")!!
        ,spf.getString("description","")!!
        ,skills
        ,experiences
        ,educations
        ,certifications
        ,spf.getString("template","")!!
        ,sImage)
    val gson= Gson()
    val value=gson.toJson(cv)
    Log.i( tag: "TEST2",value)
    var call = APIClient.apiInterface().postCV(cv)
```

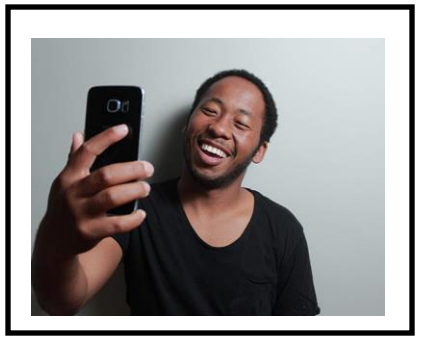
MAJOR COMPONENTS

Oracle Cloud

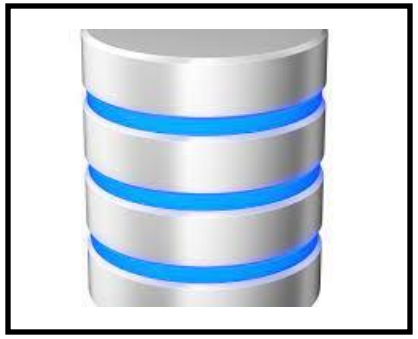


WebView

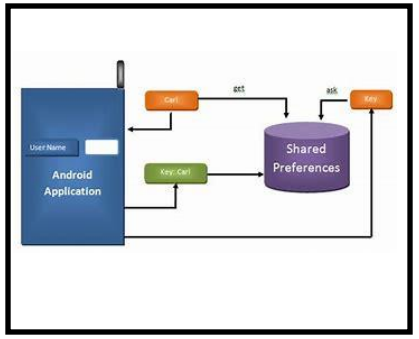
Recycler View



Multimedia

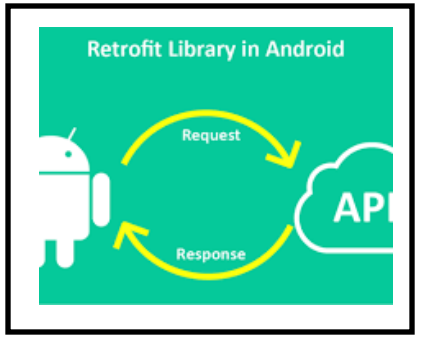


Room DB



Shared-Preferences

Navigation Graph



GSON/Retrofit

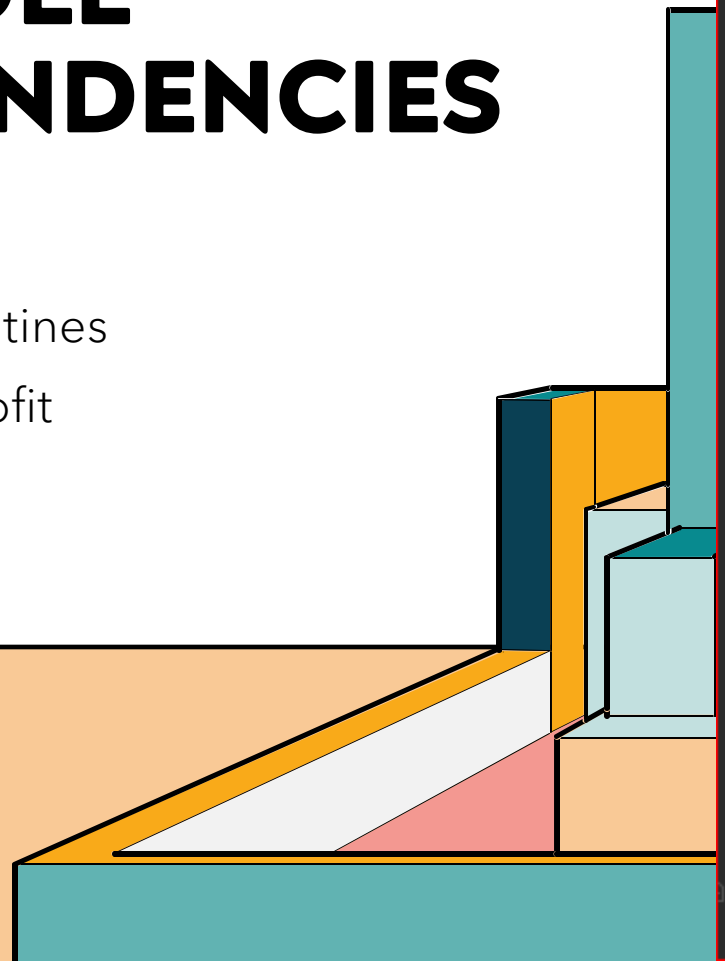
GRADLE DEPENDENCIES

-Room DB

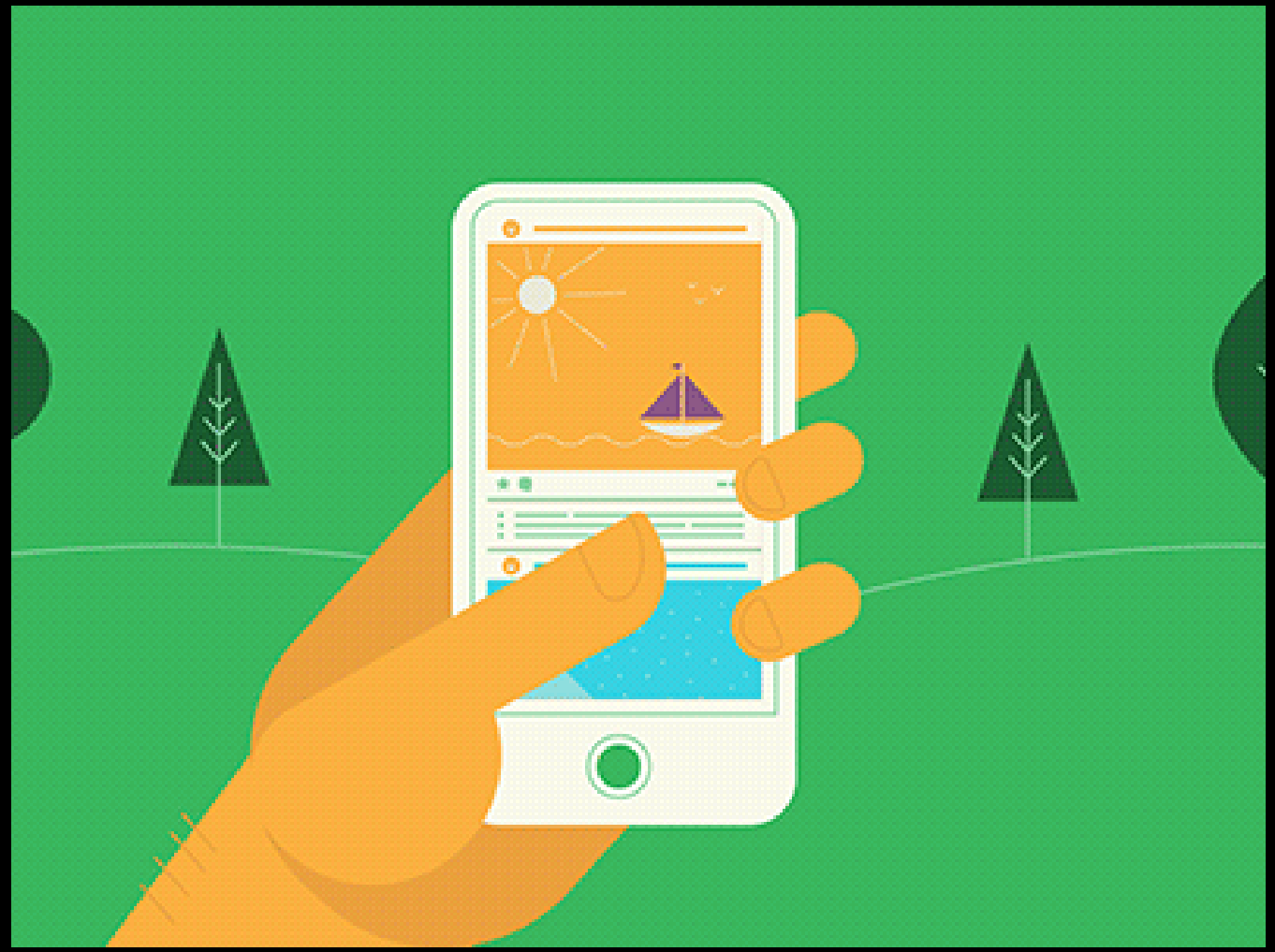
-Kotlin Coroutines

-GSON/Retrofit

-Navigation



```
dependencies {
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'


    // version of the room
    def room_version :String = "2.4.3"
    // Room Library
    implementation "androidx.room:room-runtime:$room_version"
    kapt "androidx.room:room-compiler:$room_version"

    // Kotlin Coroutines
    implementation "androidx.room:room-ktx:$room_version"
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.0'

    //GSON retrofit
    implementation 'com.google.code.gson:gson:2.9.1'
    implementation 'com.squareup.retrofit2:retrofit:2.6.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.6.0'
    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.5.1'
    implementation 'com.google.android.material:material:1.6.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.5.1'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.5.1'

    //navigation
    implementation 'androidx.navigation:navigation-fragment-ktx:2.5.2'
    implementation 'androidx.navigation:navigation-ui-ktx:2.5.2'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```



LIVE DEMO

SCI

- Android JetPack is a set of libraries designed to work efficiently, to write less boilerplate code and to improve the quality of the applications. Similarly, practicing Transcedental Meditation regularly there is an improvement in many aspect of our life, we work more efficiently improving quality of life.

SCI

- Kotlin is an expressive language, very easy and intuitive to learn, designed to help the process of software development.
  Transcendental Meditation is also a technique very easy to learn and practice, designed to improve our life by releasing stress and other anomalies.

GITHUB REFERENCE

HTTPS://GITHUB.COM/OSCAR-GORDILLO/CV_BUILDER_2.GIT

THANK YOU