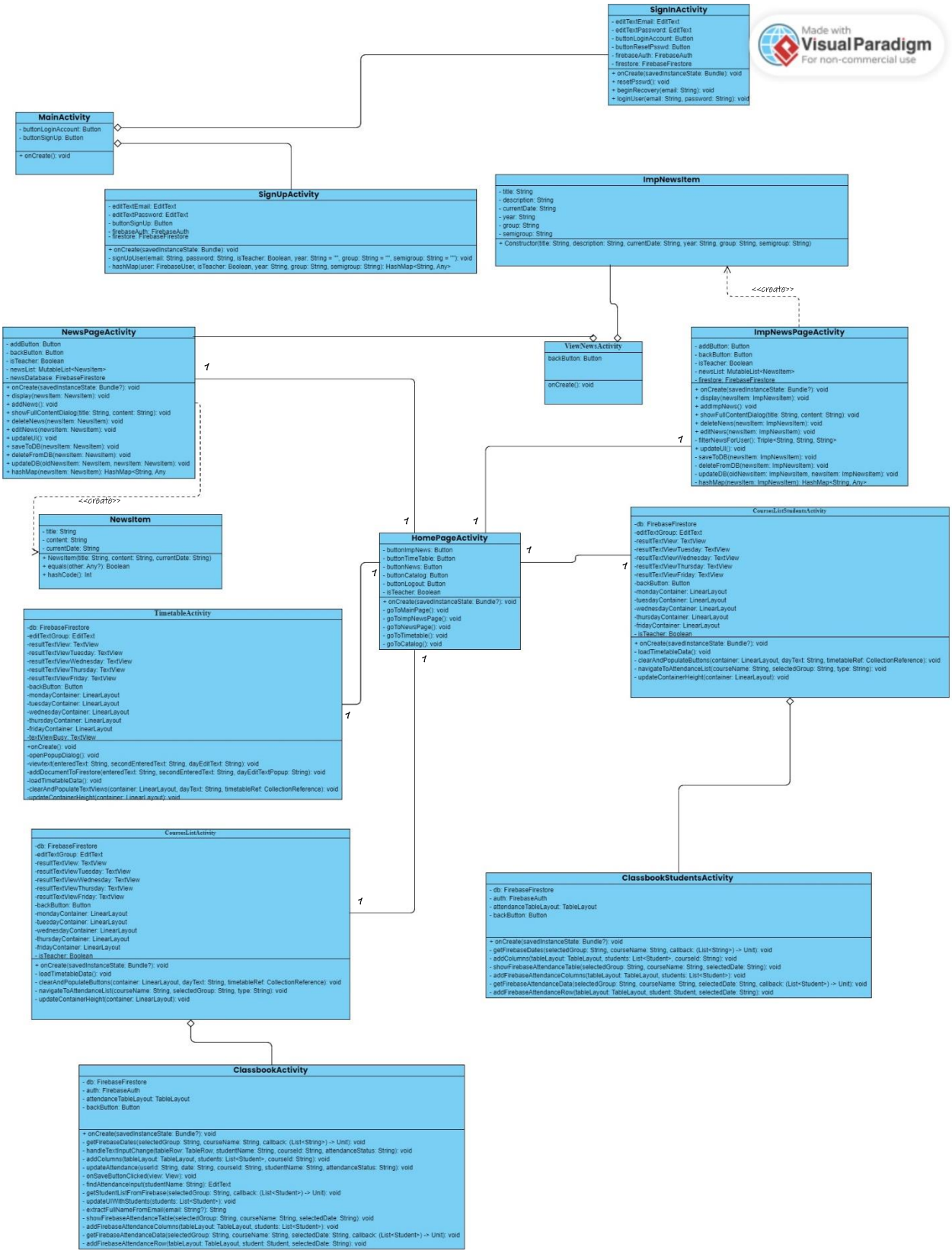
Die Anforderungen

Als kurze Beschreibung des Projekts schlage ich vor, eine Anwendung zu erstellen, die die verschiedenen Bedürfnisse der Studierenden automatisiert. Die Hauptfunktionalität besteht darin, den Stundenplan nach Fakultät, Gruppe und Halbgruppe anzuzeigen, damit der Stundenplan leicht verständlich ist. Es basiert auch auf einer Büro-Social-Media-Anwendung, bei der jeder Schüler oder Lehrer sein eigenes, einzigartiges Profil hat und den Stundenplan des anderen einsehen kann, mit dem Hinweis, dass man persönliche Veranstaltungen zur Privatisierung hinzufügen kann. Darüber hinaus können Lehrer jederzeit mit einer einzigen Schaltfläche die Anwesenheitsliste erstellen und Ankündigungen für alle Schüler oder für eine bestimmte Gruppe von Schülern veröffentlichen. Daher sehen Studierende ihre Abwesenheiten und Anwesenheiten und erhalten Benachrichtigungen basierend auf den Ankündigungen.

Als Merkmale sind: Sign-Up Funktion, Log In Funktion, Nachrichtenseite, Wichtige Nachrichtenfunktion, Fahrplanseite, Klassenbuchseite

Die Architektur



Der Quellcode

```
    private fun clearAndPopulateTextViews(container: LinearLayout, dayText:
String, timetableRef: CollectionReference) {
        container.removeAllViews()

        val dayTextView = TextView(this)
        dayTextView.text = dayText
        dayTextView.textSize = 30f
        //dayTextView.gravity = Gravity.CENTER
        container.addView(dayTextView)

        timetableRef.get()
            .addOnSuccessListener { result ->
                for (document in result) {
                    val stringBuilder = StringBuilder()

                    val courseName = document.id
                    val hours = document.getString("Hours")
                    val room = document.getString("Room")
                    val teacher = document.getString("Teacher")
                    val type = document.getString("Type")
                    val week = document.getString("Week")

                    // Create a SpannableString for the entire text
                    val fullText = "$hours, $week\n$courseName,
$type\n$teacher\n$room"
                    val spannableString = SpannableString(fullText)

                    // Set the text size for the "Hours" part
                    if (hours != null) {
                        val hoursIndex = fullText.indexOf(hours)
                        spannableString.setSpan(
                            AbsoluteSizeSpan(20, true),
                            hoursIndex,
                            hoursIndex + hours.length,
                            Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
                        )
                    }

                    // Set the text size for the "Week" part
                    if (week != null) {
                        val weekIndex = fullText.indexOf(week)
                        spannableString.setSpan(
                            AbsoluteSizeSpan(20, true),
                            weekIndex,
                            weekIndex + week.length,
                            Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
                        )
                    }

                    // Set the text size for the "Teacher" part
                    if (teacher != null) {
                        val teacherIndex = fullText.indexOf(teacher)
                        spannableString.setSpan(
                            AbsoluteSizeSpan(20, true),
```

```
                            teacherIndex,
                            teacherIndex + teacher.length,
                            Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
                        )
                    }

                    // Set the text size for the "Room" part
                    if (room != null) {
                        val roomIndex = fullText.indexOf(room)
                        spannableString.setSpan(
                            AbsoluteSizeSpan(20, true),
                            roomIndex,
                            roomIndex + room.length,
                            Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
                        )
                    }

                    // Log values for debugging
                    Log.d("TimetableActivity1", "Course: $courseName")
                    Log.d("TimetableActivity1", "Hours: $hours")
                    Log.d("TimetableActivity1", "Room: $room")
                    Log.d("TimetableActivity1", "Teacher: $teacher")
                    Log.d("TimetableActivity1", "Type: $type")
                    Log.d("TimetableActivity1", "Week: $week")

                    // Build a string with the retrieved data
                    stringBuilder.append("$hours, $week\n")
                    stringBuilder.append("$courseName, $type\n")
                    stringBuilder.append("$teacher\n")
                    stringBuilder.append("$room")

                    // Create a new TextView for each document
                    val courseTextView = TextView(this)

                    // Set the entire spannable string to the TextView
                    courseTextView.text = spannableString

                    courseTextView.textSize = 24f // Set the text size to
20sp
                    courseTextView.gravity = Gravity.CENTER // Center the
text horizontally

                    // Set a custom background with rounded corners and
different colors for each course
                    val colorArray = arrayOf("#0189c3", "#6f3096", "#004174",
"#623d5a")
                    val randomColor = colorArray.random()
                    val drawable =
resources.getDrawable(R.drawable.rounded_rectangle, theme)

drawable.setTint(android.graphics.Color.parseColor(randomColor))

                    // Set layout parameters for width and gravity
                    val params = LinearLayout.LayoutParams(
                        LinearLayout.LayoutParams.WRAP_CONTENT,
                        LinearLayout.LayoutParams.WRAP_CONTENT
                    )
```

```
                    params.width = 850  // Adjust the width as needed
                    params.gravity = Gravity.CENTER

                    // Adjust the size of the rectangle
                    val padding =

resources.getDimensionPixelSize(R.dimen.course_padding) // Define
'course_padding' in res/values/dimens.xml
                    courseTextView.background = drawable
                    courseTextView.setPadding(padding, padding, padding,
padding)

                    // Apply layout parameters to the courseTextView
                    courseTextView.layoutParams = params

                    // Add bottom margin to create space between rectangles
                    params.bottomMargin =
resources.getDimensionPixelSize(R.dimen.course_spacing)
                    container.addView(courseTextView)
                }
            }
    }
}
```

Nein.	Prüfpunkt/Mängelerklärung	Setzen Sie ein Häkchen (□) bei der entsprechenden Spalte	
		Ja	N/A
R01	Die Anforderungen sind unvollständig.		✓
R02	Es fehlen Anforderungen .		✓
R03	Die Anforderungen sind falsch.		✓
R04	Die Initialisierung des Systemzustands wurde nicht berücksichtigt.		✓
R05	Die Funktionen sind nicht ausreichend definiert.		✓
R06	Die Bedürfnisse der Nutzer werden unzureichend angegeben.		✓
R07	Kommentare.	✓	

Nein.	Prüfpunkt/Mängelerklärung	Setzen Sie ein Häkchen (□) bei der entsprechenden Spalte	
		Ja	N/A
A01	Ist die Gesamtorganisation des Programms klar, einschließlich guter Architektonischer Überblick?	✓	
A02	Ist die Partitionierung und Schichtung des Subsystems und des Pakets logisch folgerichtig?	✓	
A03	Berücksichtigt die Architektur alle Anforderungen?	✓	
A04	Sind die Klassen in einem Subsystem, das die identifizierten Dienste unterstützt? für das Subsystem?	✓	
A05	Gibt es eine kohärente Fehlerbehandlungsstrategie?		✓
A06	Wurden klassische Entwurfsmuster berücksichtigt, wo sie möglicherweise berücksichtigt wurden? in die Architektur integriert werden?	✓	
A07	Spiegeln der Name und die Beschreibung der einzelnen Klassen eindeutig die spielte eine Rolle ?	✓	
A08	Erfasst die Beschreibung jeder Klasse genau die Verantwortlichkeiten der Klasse?	✓	
A09	Sind die Rollennamen von Aggregationen und Zuordnungen korrekt? Beschreiben Sie die Beziehung zwischen den verwandten Klassen?	✓	
A10	Sind die Schlüsselentitätsklassen und ihre Beziehungen konsistent mit das Domänenmodell (falls vorhanden) und die Anforderungen?	✓	

Nein.	Prüfpunkt/Mängelerklärung	Setzen Sie ein Häkchen (□) bei der entsprechenden Spalte	
		Ja	N/A
C01	Die Entscheidungslogik oder -sequenzierung ist fehlerhaft oder unzureichend.		✓
C02	Die Verzweigung ist fehlerhaft.		✓
C03	Es gibt undefinierte Schleifenabschlüsse.		✓
C04	Es liegen E/A-Formatfehler vor.		✓
C05	Aufrufe von Unterprogrammen werden verletzt.		✓
C06	Es gibt Fehler bei der Aufbereitung oder Verarbeitung von Eingabedaten.		✓
C07	Es liegen Fehler bei der Ausgabeverarbeitung vor.		✓
C08	Fehler bei der Verarbeitung von Fehlermeldungen liegen vor.		✓
C09	Es gibt Verwirrung bei der Verwendung von Parametern.		✓
C10	Es gibt Fehler in Schleifenzählern.		✓
C11	Beim Schreiben von Variablennamen werden Fehler gemacht.		✓
C12	Variablentyp und -dimensionen werden falsch deklariert.		✓