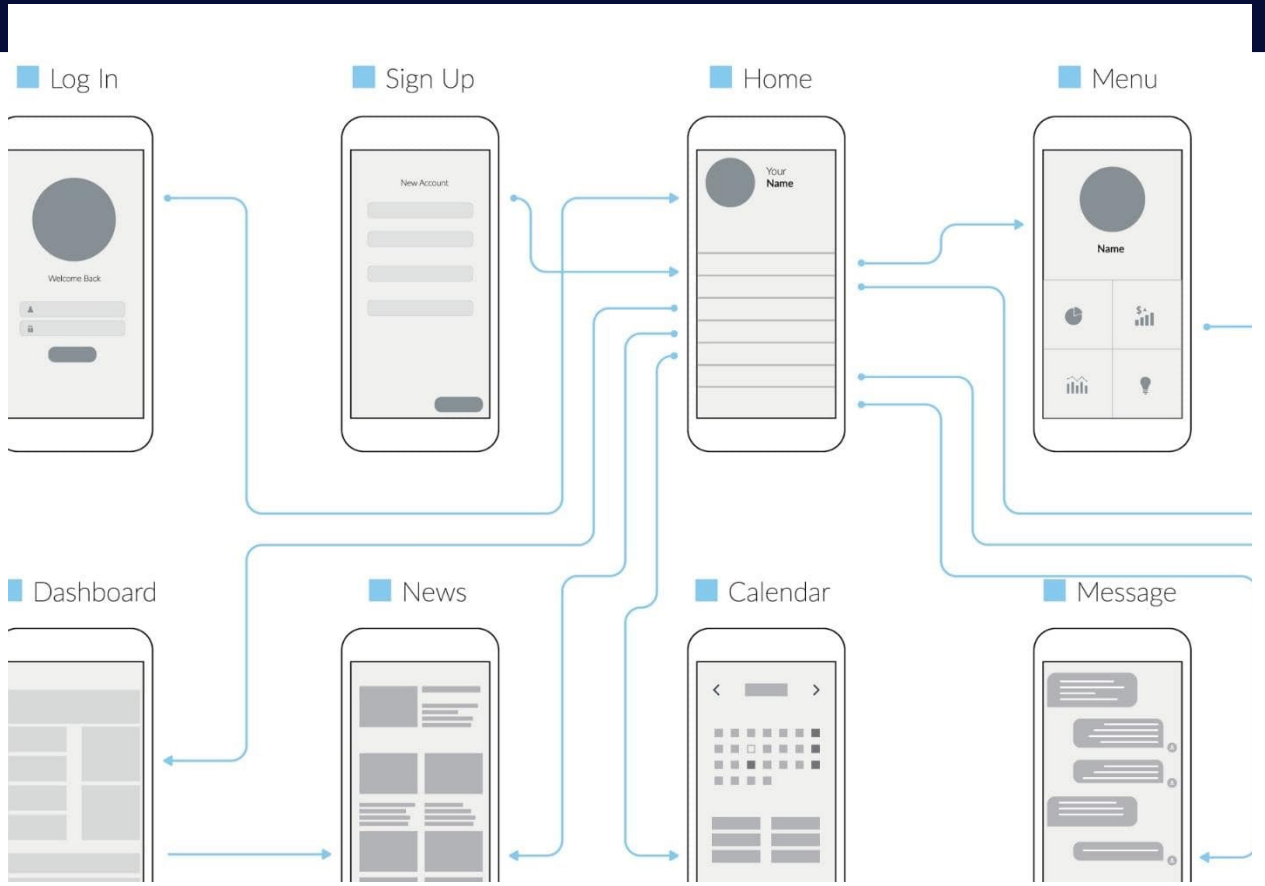
Syntax Institut

Modul 3 – Android App Entwicklung mit Kotlin

Activities und Intents

Gliederung

- Was sind Intents?
- Unterschied implicit und explicit
- Wie werden sie programmiert?



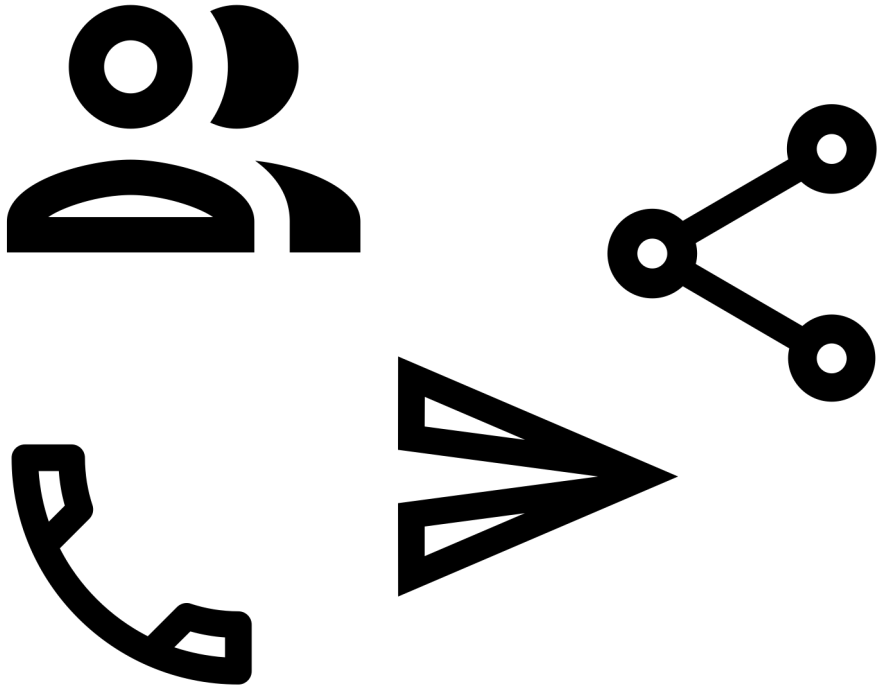
Quelle: https://www.leanplum.com/blog/user-flow/

Was sind Intents?

intent – „Absicht“

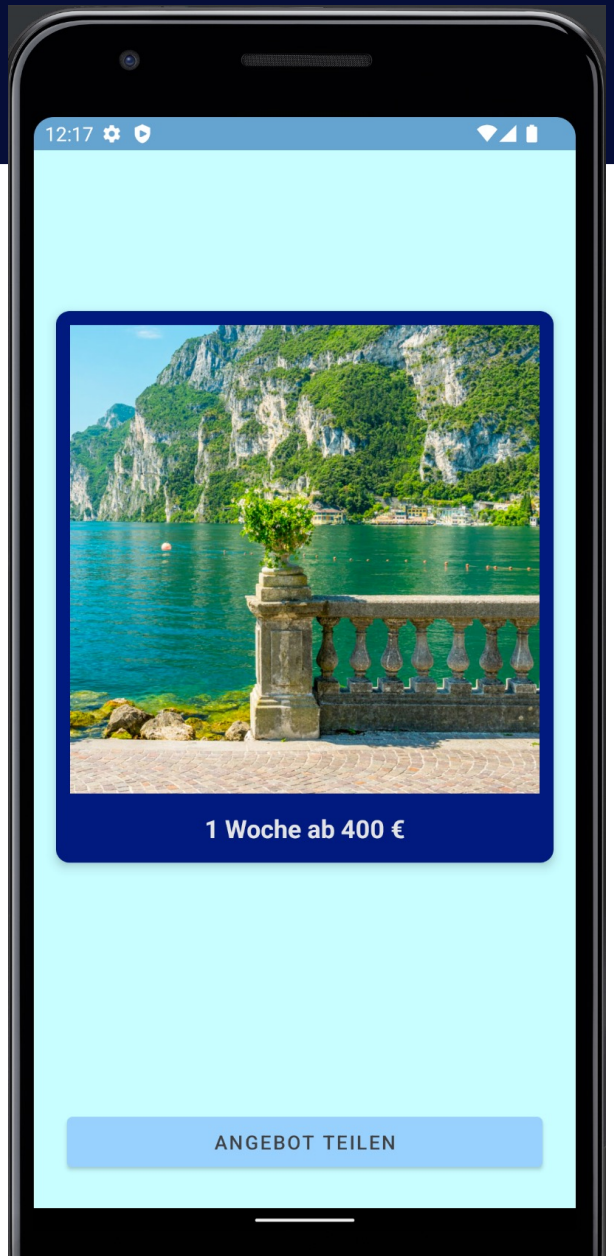
Objekte die andere Komponenten auffordern etwas zu machen

meist (aber nicht nur) zum Starten von Activities verwendet



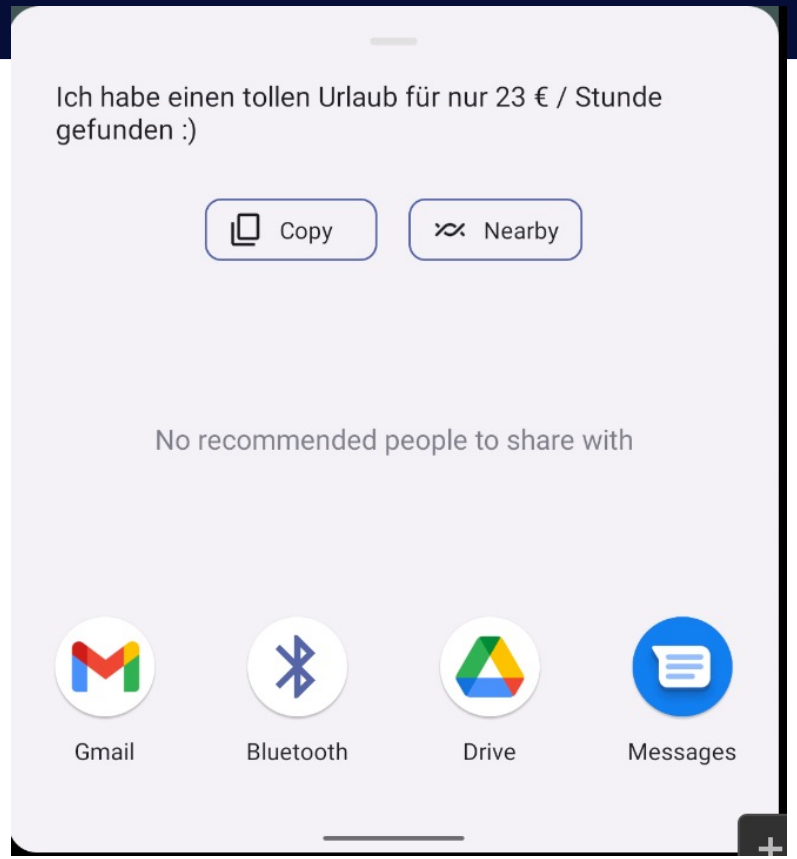
Explicit Intent

- sehr spezifische Absicht
- man weiß genau welche Activity man aufrufen möchte
- z.B.: anderer Screen derselben App



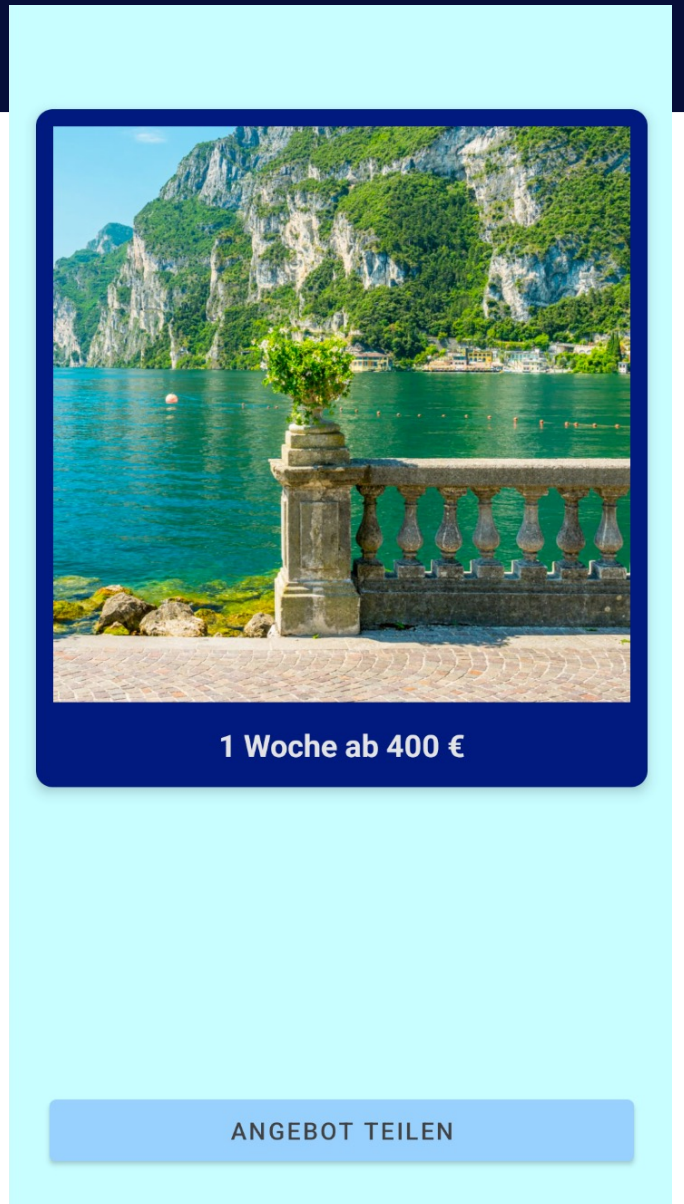
Implicit Intent

- abstrakte Absicht
- Betriebssystem sucht die passende Activity
- z.B.: Email senden



Beispiel: DetailScreen für Reisebüro

- Explicit Intent
- klick auf Reiseziel öffnet DetailActivity mit vergrößerter Ansicht



ItemAdapter.kt

- Verweis auf CardView damit die gesamte View klickbar ist

- bei Klick wird ein Intent konstruiert mit dem Ziel eine DetailActivity zu starten

- damit die DetailActivity weiß was sie anzeigen soll bekommt sie Verweise auf Text und Bild des Angebots als Extra mitgeliefert

```
// der Adapter braucht den Context um auf String Resourcen zuzugreifen
// und die Liste an Jokes um sie für die RecyclerView vorzubereiten
class ItemAdapter(...) : RecyclerView.Adapter<ItemAdapter.ItemViewHolder>() {

    // der ViewHolder weiß welche Teile des Layouts beim Recycling angepasst werden
    class ItemViewHolder(val view: View) : RecyclerView.ViewHolder(view) {
        val textView: TextView = view.findViewById(R.id.item_text)
        val imageView: ImageView = view.findViewById(R.id.item_image)
        val cardView: CardView = view.findViewById(R.id.item_cardView)
    }

    // hier werden neue ViewHolder erstellt
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ItemViewHolder {...}

    // hier findet der Recyclingprozess statt
    // die vom ViewHolder bereitgestellten Parameter werden verändert
    override fun onBindViewHolder(holder: ItemViewHolder, position: Int) {
        val item = dataset[position]
        holder.textView.text = context.resources.getString(item.stringResource)
        holder.imageView.setImageResource(item.imageResource)
        holder.cardView.setOnClickListener {  it: View!
            val context = holder.view.context
            val intent = Intent(context, DetailActivity::class.java)
            intent.putExtra( name: "imageId", item.imageResource)
            intent.putExtra( name: "stringId", item.stringResource)
            context.startActivity(intent)
        }
    }

    // damit der LayoutManager weiß wie lang die Liste ist
    override fun getItemCount(): Int {...}
}
```

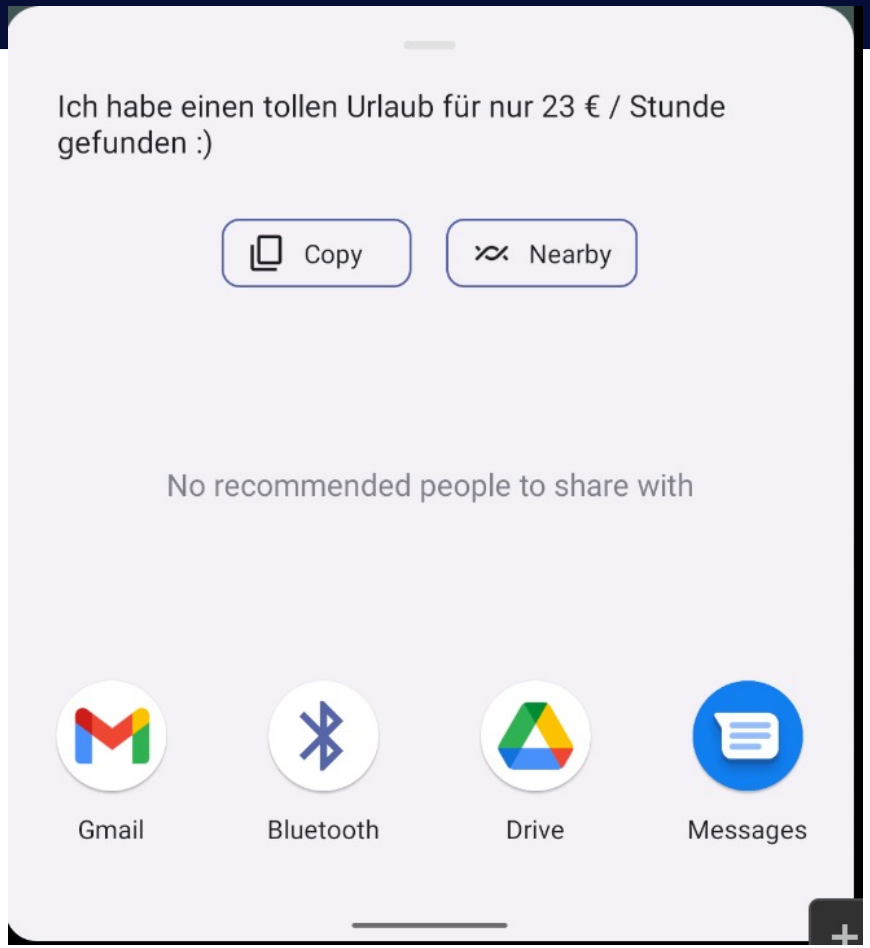
DetailActivity.kt

- Verweise werden aus den Extras geladen
- Mit Hilfe der Verweise werden die richtigen Resources in die Views geholt

```
class DetailActivity : AppCompatActivity() {

    lateinit var binding: ActivityDetailBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = DataBindingUtil.setContentView( activity: this, R.layout.activity_d

        val stringId = intent.extras?.getInt( key: "stringId")
        val imageId = intent.extras?.getInt( key: "imageId")

        var detailText = ""

        if (stringId != null) {
            detailText = getString(stringId)
            binding.detailText.text = detailText
        }
        if (imageId != null) {
            binding.detailImage.setImageResource(imageId)
        }
```

Beispiel: ShareButton für Reisebüro

- Implicit Intent
- klick auf Button öffnet Dialog zum Teilen des Reiseangebots



DetailActivity.kt

- bei Klick wird ein Sendeintent erstellt
- Extra enthält Text über das Angebot
- Angabe des Typs hilft Betriebssystem beim Vorschlagen von Apps
- Chooser erstellt eine Auswahl

```
class DetailActivity : AppCompatActivity() {

    lateinit var binding: ActivityDetailBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = DataBindingUtil.setContentView( activity: this, R.layout.activity_detail)

        val stringId = intent.extras?.getInt( key: "stringId")
        val imageId = intent.extras?.getInt( key: "imageId")

        var detailText = ""

        if (stringId != null) {
            detailText = getString(stringId)
            binding.detailText.text = detailText
        }
        if (imageId != null) {
            binding.detailImage.setImageResource(imageId)
        }

        binding.detailShareButton.setOnClickListener {  it: View!
            val intent = Intent(Intent.ACTION_SEND)
            intent.putExtra(Intent.EXTRA_TEXT,  value: "Ich habe einen tollen Urlaub für nur $deta
            intent.type = "text/plain"
            val shareIntent = Intent.createChooser(intent,  title: null)
            startActivity(shareIntent)
        }
    }
}
```

Activities und Intents

Wiederholung - Was haben wir heute gelernt?

1	Was sind Intents?
2	Unterschied zwischen implicit und explicit
3	Wie wird sie programmiert?

Syntax Institut

Quelle: https://www.wissenschaft-x.com/a-chronological-history-of-social-media

Viel Spaß!