Syntax Institut

Modul 3 – Android App Entwicklung mit Kotlin

Wiederholung

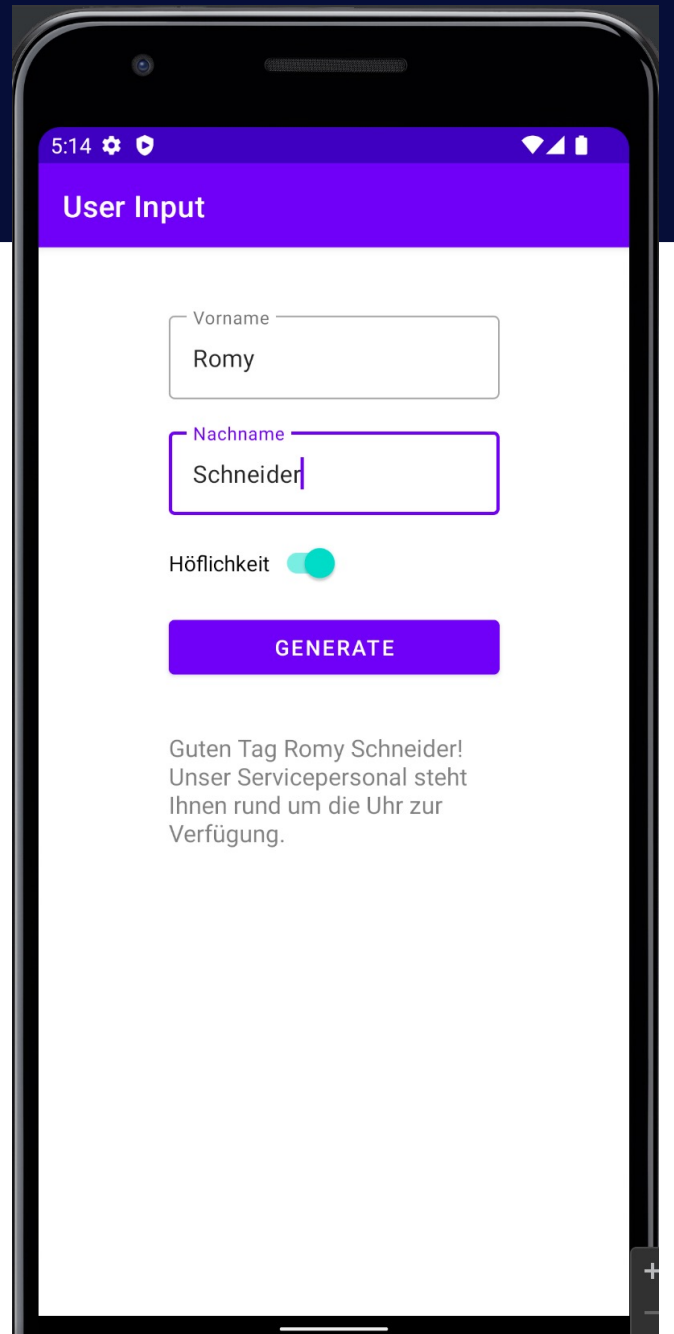
Gliederung

- Was haben haben wir bis jetzt gelernt?
- Was kommt noch?
- Offene Fragen
- Euer Feedback

Quelle: https://innovationbydesign.pressbooks.com/chapter/ideate/

User Input

- UI Elemente für Benutzereingabe
- Strings immer von Resources abrufen um Übersetzung zu ermöglichen
- Data Binding verbindet Layout und Code



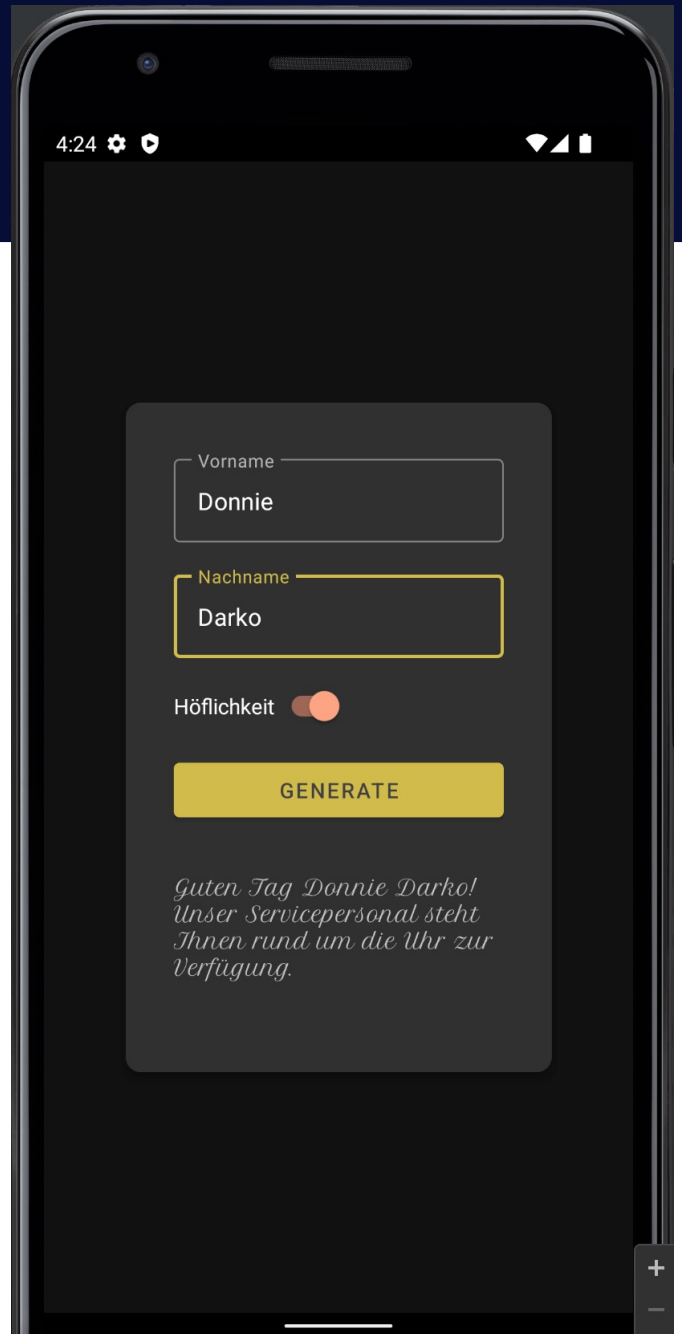
Databinding und Debugging

- DataBinding ermöglicht Zugriff auf alle Layoutelemente
- Mittels Log können Infos in der Console ausgegeben werden
- Exception Handling fängt Fehler ab und verhindert den Absturz der App

```
private const val TAG = "MainActivity"

class MainActivity : AppCompatActivity() {

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

        // setContentView(R.layout.activity_main)
        val binding: ActivityMainBinding =
            DataBindingUtil.setContentView( activity: this, R.layout.activity_main)

        Log.v(TAG,  msg: "Allgemeiner Log")

        Log.d(TAG,  msg: "Debug Log")

        Log.i(TAG,  msg: "Relevanter Log")

        Log.w(TAG,  msg: "Noch nicht problematischer Log")

        Log.e(TAG,  msg: "Fehler Log")

        Log.wtf(TAG,  msg: "unvorstellbar problematischer Log")
```

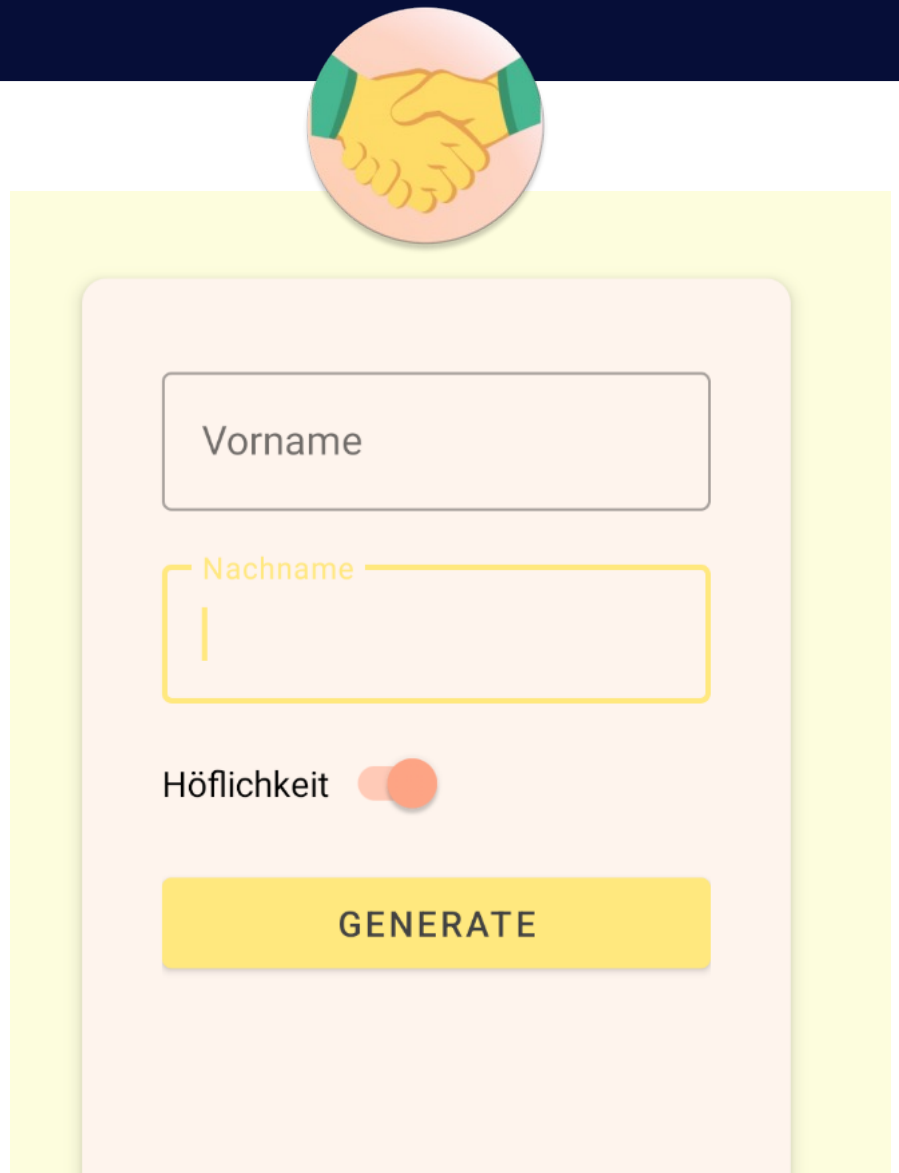
App Themes

- **Themes** bestimmen das grundlegende Erscheinungsbild der App (Farben, Schriften, usw)
- Farben sind in res/values/**colors.xml** definert
- Ein dunkles Theme wird in einem seperaten **values-night** Ordner erstellt



App Icon

- Ein **App-Icon** kann im Image Asset Studio erstellt werden
- Ein dynamisches Icon besteht aus Vordergrund und Hintergrund
- **Material Design Components** bietet anpassbare, fertige Bausteine für die UI Gestaltung



RecyclerView

- Code wird in **Packages** organisiert
- **Data Class** hilft beim abspeichern komplexerer Datensätze
- **Data Source** bereitet Daten einheitlich für die App vor
- in **list_item.xml** wird das Aussehen eines Listenelements bestimmt
- Der RecyclerView **Adapter** kümmert sich mit hilfe des **ViewHolders** ums Recycling der Views

Was kommt noch?

- Mehrere Screens und Navigation
- MVVM Architektur Muster
- Live Data
- Coroutines
- Webservices und API Calls
- Caching Data

Quelle: https://www.asfinag.at/verkehr-sicherheit/tunnelsicherheit/

Offene Fragen

Quelle: https://healthblog.uofmhealth.org/cancer-care/what-did-my-doctor-say-what-to-do-when-you-dont-understand

Feedback

Quelle: https://dubsism.files.wordpress.com/2015/08/austin-powers-feedback-meme.jpeg

Syntax Institut

Gut gemacht!

Quelle: https://spotlightenglish.com/culture/history-of-applause/