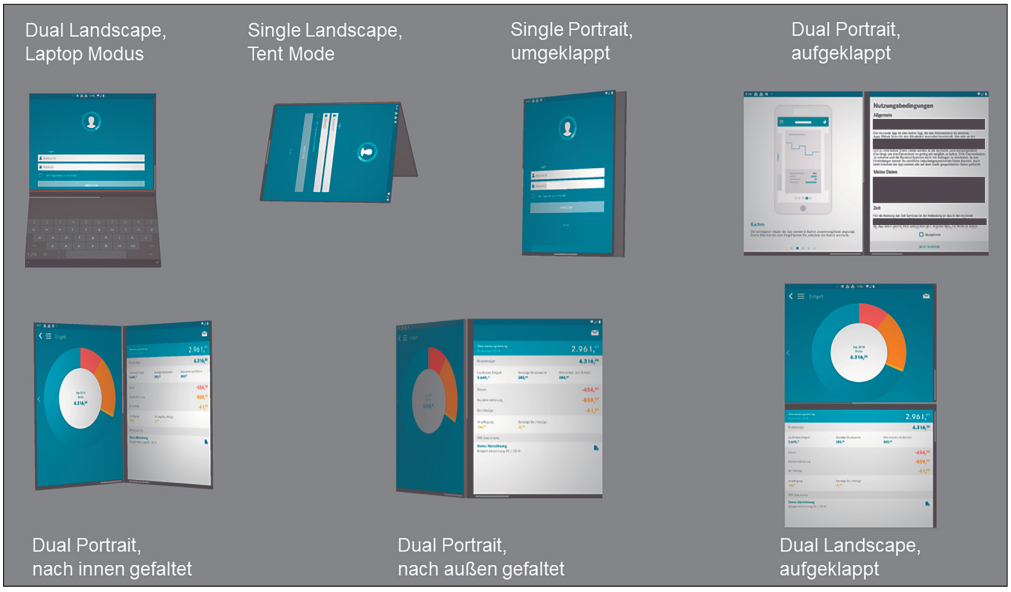
APP-ENTWICKLUNG

Eine App, zwei Bildschirme

Tipps und Tricks zur Entwicklung für Dual-Screen-Geräte.

People need their phones to be productive – but phones have their limits" – mit diesen Worten stellte der damalige Chief Product Officer der Microsoft Gerätesparte Panos Panay im Oktober 2019 erstmals das Surface Duo vor – ein faltbares Smartphone mit zwei getrennten Bildschirmen [1]. Mit seinem sichtbaren Scharnier bildet es einen wesentlichen Kontrast zu bereits am Markt verfügbaren Foldables, wie beispielsweise dem Samsung Galaxy Z Fold2 [2].

Die Vision dahinter: Durch zwei unabhängig voneinander bedienbare Screens können Nutzer mehrere Applikationen beziehungsweise Funktionen einer App gleichzeitig verwenden, ohne dabei die Ansicht wechseln zu müssen. Etwa, indem sie ihren Terminkalender durchsuchen, während sie eine E-Mail schreiben.



Mögliche Positionen des Surface Duo (Bild 1)

Die Darstellung von Apps ist auf einem oder aufgespannt über beide Bildschirme möglich. Die Bildschirme selbst können um 360 Grad gedreht und damit in jedem Winkel betrachtet werden [3], siehe Bild 1. Das so entstehende Flow-Erleben soll die Produktivität erheblich steigern. Oder, wie es Panos Panay ausdrückt: „We absolutely know scientifically that you will be more productive on two screens – much more than one screen ever could do."

Doch was bedeutet das für die App-Entwicklung? Wie können Anwendungen so gestaltet werden, dass sie möglichst große Produktivitätspotenziale für die User freisetzen?

Top-3-Tipps für die Dual-Screen-App-Entwicklung

- **Vor dem Start:** Codebasis sowie Unterstützungsmöglichkeiten der eingesetzten Bibliotheken analysieren und Aufwand für Kompatibilitätsanpassung schätzen.
- **Schrittweises Vorgehen:** Zunächst die Kompatibilität sicherstellen und damit eine fundierte Basis für weitere Anpassungen legen.
- **Ausschöpfen der Möglichkeiten:** Die Anpassungsmöglichkeiten sind vielfältig; die Umsetzung kann iterativ erfolgen und parallelisiert werden.

Zur Beantwortung dieser Fragen beschreibt der Beitrag zunächst die Möglichkeiten der Android-Entwicklung für Dual-Screen-Geräte, bevor anschließend zentrale Erkenntnisse aus der Anpassung einer App für das Surface Duo zusammengefasst werden.

Als Referenz diente dabei die interne Mitarbeiter-App eines Großkonzerns, die Funktionalitäten wie Zeiterfassung, das Einsehen von Gehaltsnachweisen oder digitale Krankmeldungen umfasst.

Dual-Screen-Frameworks

Native Anwendungen, welche mithilfe des Android-Frameworks entwickelt wurden, können unter Verwendung des Surface Duo SDK von Microsoft [4] oder des Window Manager API von Android [5] an Dual-Screen-Geräte angepasst werden.

Zusätzlich existiert für folgende weitere Frameworks Dual-Screen-Unterstützung:

- React Native [6],
- Flutter [7],
- Xamarin [8],
- CSS-Media-Queries zur Erkennung des zweiten Bildschirms und des Scharniers [9],
- Unity [10].

Die Funktionen der oben genannten Frameworks ermöglichen es, Anwendungen so anzupassen, dass sie für Dual-

Screen-Geräte optimiert sind und gleichzeitig weiterhin auf Single-Screen-Geräten nutzbar bleiben. Eine zweite Codebasis ist nicht notwendig.

Hybrides Vorgehen für die Dual-Screen-Entwicklung

Aufgrund der spezifischen technischen Anforderungen von Dual-Screen-Apps erfordert deren Entwicklung ein angepasstes Entwicklungsmodell (vergleiche Bild 2). Hierbei muss zunächst die Kompatibilität der Anwendung mit Dual-Screen-Geräten sichergestellt werden (Phase 1, P1).

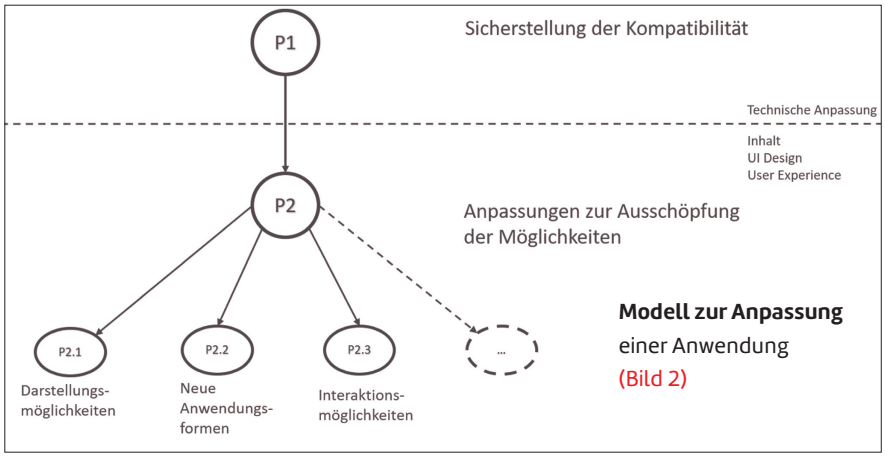
Während die Ergebnisse dieses Schritts nach außen hin kaum sichtbar sind, bilden sie dennoch die zentrale Basis für die weiteren Entwicklungsschritte. Deshalb muss Phase 1 vor Beginn von Phase 2 (P2) abgeschlossen sein, während die Unterschritte in Phase 2 nicht ihrer Reihenfolge nach abgearbeitet werden müssen. Hier sind mehrere Iterationen oder Parallelisierung denkbar.

Phase 1 – Kompatibilität bestehender Apps

Vor dem Start der Entwicklung sollten die vorhandene Codebasis und die Dual-Screen-Unterstützung der verwendeten Frameworks analysiert werden. Dies dient als Grundlage für die Aufwandsschätzung von Phase 1.

Dokumentationen von Android [11] und die Anpassung der Anwendung haben gezeigt, dass folgende Punkte beim Sicherstellen der Kompatibilität berücksichtigt werden sollten:

- Erlauben aller Gerätepositionen: Sind Orientierungsänderungen (*screenOrientation* auf *portrait* oder *landscape* fixiert) und/oder Größenänderungen (*resizeableActivity* auf *false*) nicht erlaubt, dann können sich Anwendungen bei Auf- und Zuklappen des Geräts nicht anpassen. Den entsprechenden Ausschnitt aus dem Android-Manifest zeigt Bild 3.
- Kontinuität der Anwendung: Orientierungs- und Größenänderungen sind bei Android Konfigurationsänderungen und haben deshalb einen Neustart der Activities zur Folge. Durch eine klare Trennung von Oberfläche und Datenhaltung muss dies ohne Datenverlust möglich sein (MVVM-Pattern). Der Zustand der Anwendung sollte nach einem Neustart noch derselbe wie davor sein. Standardbibliotheken von Android, etwa Android ViewModels, bringen Funktionalitäten zur Erfüllung dieser Anforderung bereits mit [12].
- Lesbarkeit der Inhalte: Durch das Scharnier in der Mitte von Dual-Screen-Geräten können Elemente verdeckt werden. Damit eine Anwendung nutzbar ist, muss sichergestellt sein, dass interaktive Elemente sichtbar bleiben. Die naheliegende Möglichkeit wäre in solch einem Fall, das Spannen über beide Bildschirme zu verbieten. Da dies nicht möglich ist, kann die entsprechende Ansicht mithilfe des „Surface Duo Frame Layouts“ (Bild 4) im aufgespannten Zustand nur auf dem linken oder rechten Bildschirm angezeigt werden [13].
- Parallelbetrieb von Anwendungen: Seit Android 10 befinden sich sämtliche angezeigten Activities im *resumed*-State. Der gleichzeitige Zugriff auf geteilte Ressourcen muss abgesichert sein (zum Beispiel Zugriff auf die Kamera). Funktionen wie Drag-and-drop und das Öffnen mehrerer Instanzen einer Anwendung sollten ebenfalls betrachtet werden.
- Nach Anpassung der Anwendung darf die Nutzung auf herkömmlichen Smartphones nicht beeinträchtigt sein.



Modell zur Anpassung einer Anwendung (Bild 2)

Phase 2 – Anpassungsvarianten

Neben der Anpassung des UI an zwei Bildschirme ergeben sich durch das Scharnier auch neue Anwendungsformen und Interaktionsmöglichkeiten. ▶

```
<manifest ...>
  ...
  android:resizeableActivity="true"
  ...
  android:screenOrientation="fullUser"
  ...
</manifest>
```

Auszug aus dem Android Manifest – Erlauben aller Gerätepositionen (Bild 3)

```
<com.microsoft.device.dualscreen.layouts.SurfaceDuoFrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:display_position="start">

    <androidx.constraintlayout.widget.ConstraintLayout ...>

</com.microsoft.device.dualscreen.layouts.SurfaceDuoFrameLayout>
```

Surface Duo Frame Layout – Anzeige einer View auf dem linken Bildschirm (Bild 4)

Anpassung des UI

Zur Umsetzung der beschriebenen Anpassungen wurden im aktuellen Beispiel das Surface Duo Layout [14] und das Surface Duo Frame Layout [13] verwendet.

Diese Layoutbibliotheken aus dem Surface Duo SDK bilden Wrapper um bestehende Ansichten, weshalb nur geringfügige Anpassungen von Oberflächenkomponenten und Quellcode erforderlich sind (Bild 5). Für Fallunterscheidungen im Quellcode wurden Funktionen des Screen-Manager-Objekts genutzt [15].

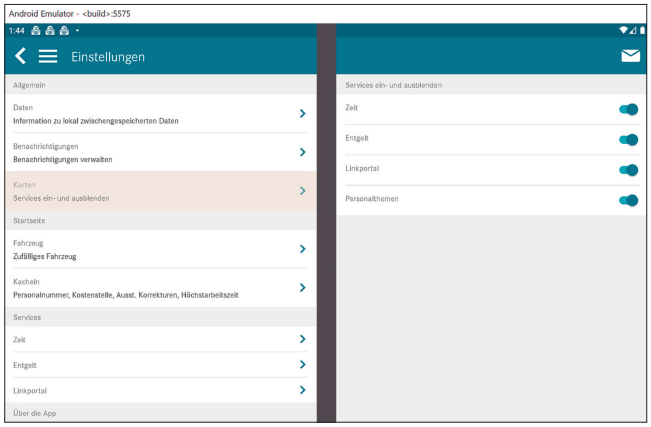
```
<com.microsoft.device.dualscreen.layouts.SurfaceDuoLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/homepage_dual_screen_wrapper"
    app:single_screen_layout_id="@layout/homepage"
    app:dual_screen_start_layout_id="@layout/homepage_dual_screen_start"
    app:dual_screen_end_layout_id="@layout/homepage_dual_screen_end" />
```

Surface Duo Layout – Wrapper um die Homepage (Bild 5)

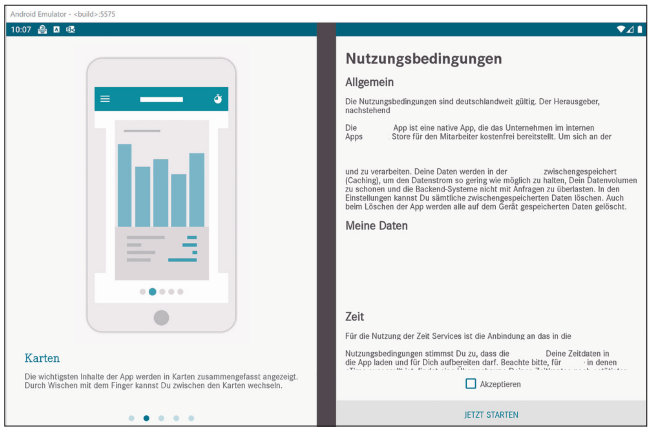
Zur Anpassung des UI bietet es sich an, bestehende Patterns zu identifizieren und diese entsprechend auf zwei Bildschirme aufzuteilen.

Die folgenden Szenarien eignen sich zur Ansicht auf einem Dual-Screen-Gerät [16] [17]:

- Das klassische List-Detail Pattern (zum Beispiel Einstellungsseiten, Mails), eine beispielhafte Umsetzung sehen Sie in Bild 6.
- Bearbeitungswerkzeuge zu Grafiken oder Texten (zum Beispiel Bildbearbeitung in der Galerie).
- Aufspannen des Inhalts für positionierbare Ansichten (zum Beispiel Kartenansichten).
- Anzeige des Inhalts in textueller und grafischer Form (zum Beispiel Grafiken und Zahleninfos).
- Leerlassen eines Bildschirms bei Ansichten mit wenig Inhalt (zum Beispiel Login-Seiten).
- Unabhängige Inhalte (zum Beispiel verschiedene Karten, Seiten eines Buches) – Two-Page Pattern, siehe Bild 7.



List-Detail Pattern, Beispielumsetzung (Bild 6)



Two-Page Pattern, Beispielumsetzung (Bild 7)

Anwendungsformen

Folgende neuen Anwendungsformen sind durch verschiedene Formfaktoren denkbar:

- Tent Mode – Aufstellen des Smartphones wie ein Zelt, mit beiden Bildschirmen nach außen gerichtet [18][19]. Dies bietet sich an zum Beispiel für die Präsentation von Inhalten und gleichzeitiges Ansehen privater Notizen (vergleiche Präsentationsansicht von PowerPoint). Ebenso aber auch für kompetitive Spiele auf Rundenbasis (wie zum Beispiel das altbekannte „Schiffe versenken“) oder für Multi-User-Modi.
- Aufklappen des Gerätes wie einen Laptop [19].
- 3D-Darstellungen [17].

Interaktionsmöglichkeiten

Mit dem Sensor API bietet Android die Möglichkeit, den genauen Neigungswinkel des Scharniers abzugreifen [20]. Dies ermöglicht folgende Falt-Interaktionen [17][21][22]:

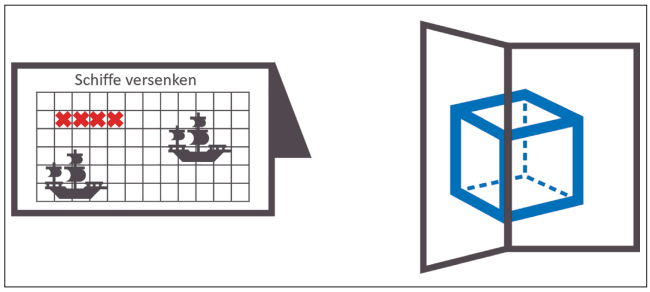
- Zoomen oder das Betätigen von Schiebereglern durch leichtes Falten. Hierbei ist zu beachten, dass dies beispielsweise mit Touch Input kombiniert werden sollte, um ungewollte Eingaben zu vermeiden.
- Ansichtswechsel beim Auf-, Zu- oder Umklappen des Geräts.

Unabhängig vom Scharniersensor ist auch eine Interaktion mit 3D-Elementen auf drei Achsen denkbar (Bild 8). Das Gerät kann hierzu im 90-Grad-Winkel nach außen aufgeklappt werden [23].

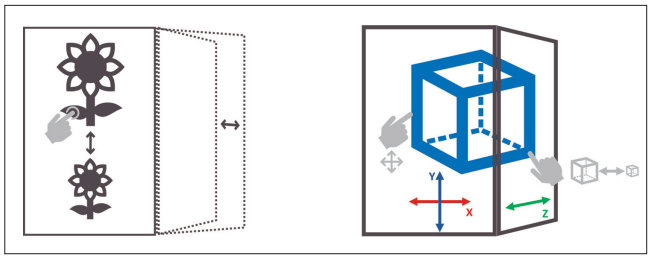
Testen

Das Surface Duo ist mittlerweile zwar in Deutschland erhältlich, mit einem Preis von circa 1500 Euro aber relativ teuer. Alternativ kann man sich mithilfe des Surface-Duo-Emulators von Microsoft auch ohne physisches Gerät an die Softwareentwicklung wagen.

Neben den zwei Bildschirmen simuliert der Emulator auch das Scharnier des Geräts und entsprechende Neigungsoperationen [24].



Schiffe versenken (links), 3D-Darstellung (rechts) (Bild 8)



Zoomen mittels Touch und des Scharniers (links), Interaktion mit einem 3D Element auf drei Achsen (rechts) (Bild 9)

Fazit und Ausblick

Dual-Screen-Geräte bieten hinsichtlich der App-Entwicklung interessante neue Möglichkeiten. Durch Bibliotheken wie die des Surface Duo SDK ist die Anpassung einer Anwendung mit geringem Aufwand möglich.

Dank der Option, die Bibliotheken des SDK mit den offiziellen APIs von Android zu verwenden, können auch Foldables wie das Samsung Galaxy Fold [2] oder das Huawei Mate Xs [25] von diesen Anpassungen profitieren. ■

[1] CNET, Surface Duo reveal presentation 2019, www.dotnetpro.de/SL2108DualScreen1

[2] Samsung, Samsung Galaxy Z Fold2 5G, www.dotnetpro.de/SL2108DualScreen2

[3] News Center Microsoft Deutschland, Microsoft Surface Duo ab heute in Deutschland, www.dotnetpro.de/SL2108DualScreen3

[4] Microsoft, Abrufen des Surface Duo SDK, www.dotnetpro.de/SL2108DualScreen4

[5] Android Developers, WindowManager, www.dotnetpro.de/SL2108DualScreen5

[6] Microsoft, React Native für Surface Duo, www.dotnetpro.de/SL2108DualScreen6

[7] Microsoft, Get Started with Flutter on Surface Duo, Surface Duo Blog, www.dotnetpro.de/SL2108DualScreen7

[8] Microsoft, Xamarin für Surface Duo, www.dotnetpro.de/SL2108DualScreen8

[9] Microsoft, Bildschirm Aufteilungs Funktion für CSS-Medien für die Dual-Screen-Erkennung, www.dotnetpro.de/SL2108DualScreen9

[10] Microsoft, Unity-Spiele für Surface Duo, www.dotnetpro.de/SL2108DualScreen10

[11] Android Developers, Building apps for foldables, www.dotnetpro.de/SL2108DualScreen11

[12] Android Developers, ViewModel Overview, www.dotnetpro.de/SL2108DualScreen12

[13] Microsoft, Surface Duo – Rahmenlayout, www.dotnetpro.de/SL2108DualScreen13

[14] Microsoft, Surface Duo – Layout, www.dotnetpro.de/SL2108DualScreen14

[15] Microsoft, Surface Duo – Bildschirm-Manager, www.dotnetpro.de/SL2108DualScreen15

[16] Microsoft, Einführung in Doppelbildschirmgeräte, www.dotnetpro.de/SL2108DualScreen16

[17] Mohammadreza Khalilbeigi, Roman Lissermann, Wolfgang Kleine, Jürgen Steimle, FoldMe: Interacting with Double-sided Foldable Displays, www.dotnetpro.de/SL2108DualScreen17

[18] Gazelle Saniee-Monfared, Kevin Fan, Qianq Xu, Sachi Mizobuchi, Lewis Zhou, Pourang Polad Irani, Wei Li, Tent Mode Interactions: Exploring Collocated Multi-User Interaction on a Foldable Device, www.dotnetpro.de/SL2108DualScreen18

[19] Ken Hinckley, Morgan Dixon, Raman Sarin, Francois Guimbretiere, Ravin Balakrishnan, Codex: a dual screen tablet computer, www.dotnetpro.de/SL2108DualScreen19

[20] Android Developers, SensorEvent, www.dotnetpro.de/SL2108DualScreen20

[21] Nicholas Chen, Francois Guimbretiere, Cassandra Lewis, Maneesh Agrawala, Enhancing Document Navigation Tasks With a Dual Display Electronic Reader, www.dotnetpro.de/SL2108DualScreen21

[22] Antonio Gomes, Roel Vertegaal, PaperFold: Evaluating Shape Changes for Viewport Transformations in Foldable Thin-Film Display Devices, www.dotnetpro.de/SL2108DualScreen22

[23] Wolfgang Büschel, Patrick Reipschläger, Raimund Dachselt, Foldable3D, in ISS'16: Proceedings of the 2016 Conference on Interactive Surfaces and Spaces : November 6–9, 2016, Niagara Falls, Ontario, Canada, 2016, pp. 367–372

[24] Microsoft, Verwenden des Surface-Duo-Emulators, www.dotnetpro.de/SL2108DualScreen23

[25] Huawei, Huawei Mate Xs, Das faltbare Displaywunder der Zukunft, www.dotnetpro.de/SL2108DualScreen24

Miriam Alber
ist Softwareentwicklerin bei der iteratec GmbH. Sie hat einen Bachelor in Wirtschaftsinformatik und studiert aktuell Informatik im Master. Schwerpunktmäßig beschäftigt sie sich mit Softwareengineering und Frontend-entwicklung.

dnpCode A2108DualScreen