2 Software im Einsatz

Ein Betriebssystem bietet dem Benutzer einerseits eine bedienbare Kommunikationsschnittstelle, auf die die hierfür passenden Programme aufsetzen, und bildet andererseits die Software-Routinen zur Hardware-Kommunikation. Jahrzehntelang war Microsoft mit einer Windows-Version unangefochtener Marktführer bei den Betriebssystemen für Personal Computer und den verwandten Notebooks. Durch die Verbreitung der Tablets und Smartphones, die die klassischen Computersysteme teilweise funktional ersetzen und auch neue Anwendungen etablieren (Apps, soziale Netze), sind insbesondere Android von Google und iOS von Apple verbreitet, die beide auf speziell angepassten Linux-Versionen basieren.

Klaus Dembowski: Smartphone-und Tablet-Hacks (ISBN-Print: 978-3-86490-423-3), 2017

2.1 Das Android-Betriebssystem

Im Jahre 2003 wurde die Firma *Android Incorporated* gegründet, deren Gründer bereits über Erfahrungen bei der Entwicklung von Betriebssystemen (DangerOS) für mobile Geräte und speziell für Smartphones verfügten. Im Laufe der Entwicklung und der Suche nach Investoren für ein neues System (Android) kam der Kontakt mit der Firma Google zustande, die Android im Jahre 2005 erwarben und es als umfassende Plattform für mobile Geräte, bestehend aus einem Betriebssystem, einer Benutzerschnittstelle und Anwendungen, propagierten.

Android wurde dann in Kooperation mit der *Open Handset Alliance* (OHA), die sich aus über 80 Softwarefirmen, Mobilfunk- und Halbleiterherstellern wie beispielsweise T-Mobile, HTC und Qualcomm zusammensetzt, kontinuierlich weiterentwickelt. Die erste Version erschien im Jahre 2008, fast gleichzeitig mit der Veröffentlichung des *Android Software Development Kit* (SDK) für Entwickler, der Freigabe als Open-Source-Plattform und des *Android Market* für den Vertrieb von Anwendungssoftware (Apps).

Seitdem werden kontinuierlich neue Android-Versionen veröffentlicht, die insbesondere unter ihren jeweiligen Code-Namen (Tabelle 2–1) bekannt sind und mittlerweile auf über 24.000 verschiedenen Gerätemodellen arbeiten. Je nach Alter und Typ des Smartphones oder Tablets ist eine bestimmte Android-Version installiert, die möglicherweise herstellerspezifische Anpassungen beinhaltet, was – wie generell der jeweilige Chipset – einen maßgeblichen Einfluss auf die Update-Möglichkeiten hat. Falls die Hersteller keine aktualisierte Android-Version für ein bestimmtes Gerät mehr auf den Markt bringen wollen, weil sie lieber neue Geräte verkaufen, besteht immer noch die Möglichkeit, hierfür eine alternative Firmware (Custom ROM) einzusetzen, um somit auch noch ältere Geräte mit aktuellen Features auszustatten, was in Kapitel 2.6 behandelt wird.

2.1.1 Versionen

Ab der Version Gingerbread (V 2.3.x) ist eine Vielzahl von Updates und Versions-Upgrades veröffentlicht worden, wobei sich einige wichtige Schritte nur in einer untergeordneten Versionszahl ausdrücken, wie beispielsweise der *USB Host-Modus*, der seit der Honeycomb 3.1-Version (10/2011) eine allgemeine Android-Unterstützung erfährt, oder *Bluetooth Low Energy* (Bluetooth Smart), welches seit der Jelly Bean-Version 4.3 (7/2013) genutzt werden kann. Mit der Android-Version *Wear* steht seit 2014 eine spezielle Version für Smartwatches und Wearables zur Verfügung und die Lollipop-Version unterstützt erstmalig 64-Bit-Architekturen.

2008	2009	2010	2011
V 1.0	V 1.1 V 1.5: Cupcake V 1.6: Donut V 2.0: Eclair	V 2.2: Froyo V 2.3.7: Gingerbread	V 3.0: Honeycomb V 4.03: Ice Cream Sandwich
2012	2013	2014	2015
V 4.1.x V 4.2.x V 4.2.1: Jelly Bean	V 4.3: Jelly Bean V 4.4.x: KitKat	V 4.4.3: KitKat V 4.4.W: Wear V 5.0.x Lollipop	V 5.1 Lollipop V 6.0 Marshmellow

Tab. 2–1 Android-Versionen im Überblick

2.1.2 Architektur und Funktionsweise

Android ist explizit für den Online-Betrieb ausgelegt, sei es mit Mobilfunk oder auch mit WLAN. Von essenzieller Bedeutung für dieses System ist außerdem, dass sich Anwendungen einfach aus dem Market laden und installieren lassen und Inhalte direkt auf dem Gerät gespeichert werden können.

Android basiert auf einem Linux-Kernel 2.6 und ist wie Apples iOS, welches ebenfalls Linux-Wurzeln aufweist, ein eigenständiges Betriebssystem, sodass es keine gemeinsame Codebasis mit Linux mehr gibt.

Android ist im Gegensatz zu iOS, welches von Apple sehr geschützt und abgeschirmt wird, immer noch ein quelloffenes System. Der Kernel ist vollständig als Open Source (GPL) lizenziert, während für die Bestandteile, die sich außerhalb des Kernels befinden, eine weniger restriktive Lizenz (z. B. LGPL oder Apache) gültig ist, die es den Herstellern erlaubt, Anpassungen und Modifikationen vorzunehmen, ohne verpflichtet zu sein, den dazugehörigen Quellcode offenlegen zu müssen.

Der Linux-Kernel bietet die grundlegende Software zum Booten und für die Hardware-Kommunikation, wie Gerätetreiber oder Funktionen für das Power Management, das durch Android-spezifische Komponenten noch erweitert wird, um den Stromverbrauch weiter senken zu können. Einige der Hardwaretreiber firmieren

unter *Closed Source*, d. h., diese Treiber sind herstellerspezifisch, etwa wie für die Camera, die Videowiedergabe oder GSM, und arbeiten außerhalb des Kernels.

Auf den Kernel setzen verschiedene Bibliotheken auf, die sowohl dem Anwender als auch dem Entwickler eine gewisse Grundfunktionalität bieten. Dazu gehören eine optimierte 3D-Grafikbibliothek (OpenGL ES), es werden Fonts (Schriften und Zeichen), Display- und verschiedene Medienfunktionen zur Verfügung gestellt sowie die SQLite-Bibliothek für Datenbankfunktionen und die WebKit-Bibliothek zum Rendern von HTML im Webbrowser. Für eine abgesicherte Kommunikation ist zudem ein *Secure Socket Layer* (SSL) standardmäßig vorhanden.

Statt einer Standard-C-Bibliothek (Glibc) wird mit Rücksicht auf einen möglichst geringen Strom- und Speicherverbrauch und eine möglichst hohe Verarbeitungsgeschwindigkeit eine spezielle – in Teilen aber Glibc-kompatible – Bibliothek verwendet (Bionic), die demgegenüber deutlich kleiner ist.

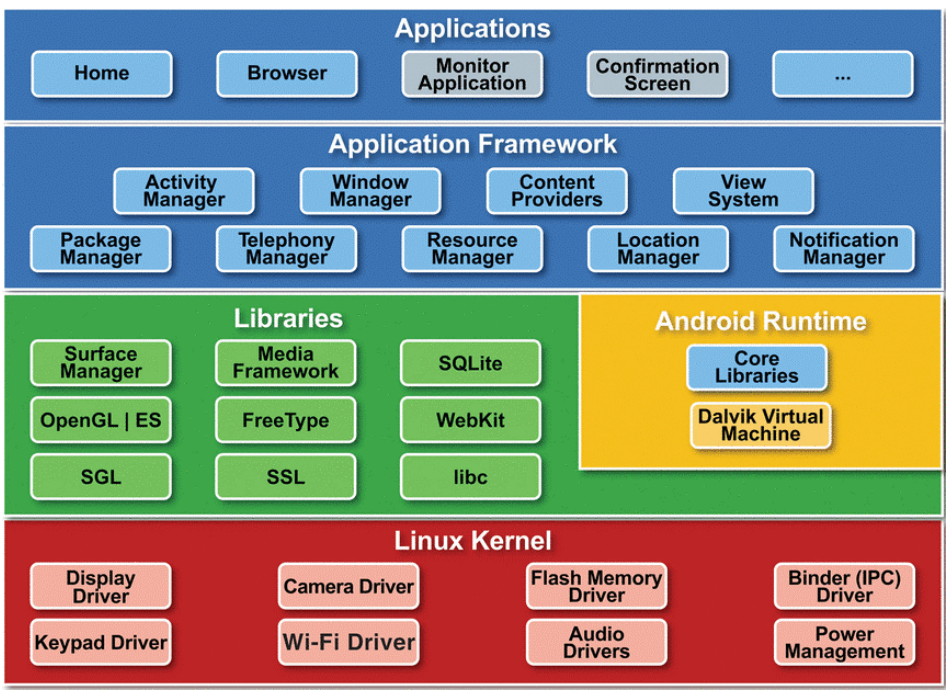

Abb. 2–1 Architektur des Android-Betriebssystems

Die für Android vorgesehenen Applikationen (Apps) und Services werden in der Programmiersprache Java geschrieben. Die Apps werden jedoch nicht als typische Java-Applikationen ausgeführt, weil Android hierfür keine entsprechenden Bibliotheken und auch keine übliche virtuelle Java-Maschine einsetzt. Stattdessen werden die Apps in der virtuellen Maschine *Dalvik* (Android Runtime) ausgeführt, die demgegenüber

im Ressourcenbedarf weit genügsamer ist, was für kleine mobile Computer ein wichtiges Kriterium ist. Des Weiteren wird dadurch auch lizenzrechtlichen Problemen aus dem Weg gegangen.

Mit Dalvik ist es möglich, verschiedene Bibliotheken mit anderen Programmiersprachen wie C/C++ entwickeln zu können, die durch Dalvik dann importiert werden können, wie es auf ähnliche Art und Weise auch mit den DLLs (Dynamic Linked Libraries) bei Windows praktiziert wird. Gleichwohl sind die Apps (*.apk) bezüglich Syntax äquivalent zu Java und vielfach ist es sogar möglich, Java-Libraries als Source-Code direkt in die Apps zu importieren. Näheres zur Applikationserstellung und Hardwarekommunikation mit Android ist in Kapitel 2.8 angegeben.

Jede Android-Anwendung läuft als ein eigener Prozess in einer separaten *Dalvik Virtual Machine*, wobei die Programme jedoch Teile anderer Anwendungen nutzen können. Wie bei anderen Betriebssystemen auch, arbeitet Android mit einer virtuellen Speicherverwaltung, bei der der physikalische Speicher über Tabellen auf virtuelle Speicheradressen abgebildet wird. Eine Anwendung arbeitet stets in einem geschützten Speicherbereich (Sandbox) und verfügt über ein eigenes Verzeichnis mit dem jeweiligen Package-Namen. Eine Besonderheit ist, dass Android für jede installierte Anwendung einen eigenen Linux-User anlegt.

Der Zugriff auf die Ressourcen erfolgt mithilfe der Application Framework-Schicht (vgl. Abbildung 2–1), die eine ganze Reihe von Managerfunktionen sowie *Content Provider* beinhaltet. Allgemein bietet das *Application Framework* dem Entwickler einen abstrakten Zugriff von den Apps aus auf die standardisiert dargestellten Systemressourcen. Verschiedene Content Provider können durch die Apps genutzt oder auch von ihnen zur Verfügung gestellt werden, beispielsweise werden die Daten des Telefonbuches über einen Content Provider für die App-Nutzung bereitgestellt.

Nachdem eine Android-App durch Java in einen Byte-Code und anschließend in einen Dalvik-Code übersetzt wurde, wird sie zusammen mit den Ressourcen (Text, Bilder, Oberfläche) in einem APK-Paket zusammengefasst, welches daraufhin installiert werden kann, was vom Package-Manager entsprechend verarbeitet wird. Jede App definiert ihre Komponenten anhand einer Manifest-Datei, die dem Package-Manager die jeweils benötigte Android-Version, die einzelnen Bestandteile der App, die notwendigen Rechte und den Ressourcenbedarf mitteilt.

Bei der Installation einer App wird von Android überprüft, ob sie über eine gültige digitale Signatur verfügt, sodass der Entwickler hiermit prinzipiell identifiziert werden kann. Für die Sicherheit der App und des dazugehörigen Schlüssels bzw. Zertifikats, welches nicht von einer Zertifizierungsstelle signiert sein muss, ist der Entwickler selbst verantwortlich. Wenn die App erfolgreich validiert worden ist, untersucht Android eine dazugehörige Konfigurationsdatei, die die Zugriffsrechte für die App bekannt gibt. Je nach Einstellung des Android-Systems und Art der App kann die Installation verweigert oder erst auf Rückfrage vom Anwender manuell gestattet wer-

den, was davon abhängt, welche Manipulationen bzw. Berechtigungen am System durch die App verursacht werden (sollen). Vielfach verlassen sich die Benutzer darauf, dass eine Warnung bei der Installation übergangen werden kann, sodass sich auf diesem Weg leicht unerwünschte und Schadsoftware einschleusen lässt.

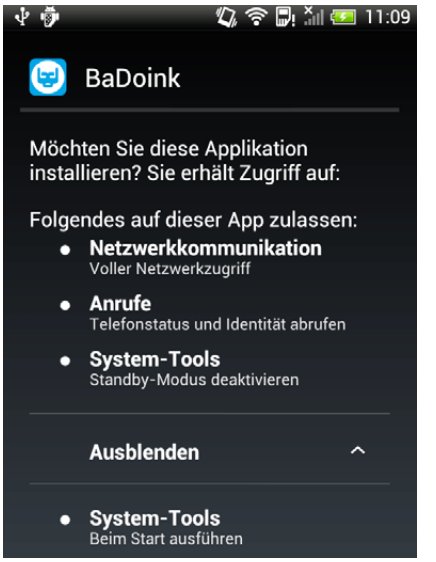

Abb. 2–2 Ob eine App derartig weitreichende Rechte benötigt, sollte stets genauer hinterfragt werden, denn es könnte (wie hier) ein Trojaner sein, der sich im System einnisten möchte.

Die Apps können sich aus verschiedenen Komponenten zusammensetzen, wobei die *Activity* die wichtigste ist, die an ein *User Interface* gebunden ist und vom Benutzer der App gestartet wird. Dieser kann eine Activity stets mit dem Back- oder auch Home-Button verschwinden lassen und zu einem anderen Zeitpunkt wieder hervorholen. Android sorgt automatisch dafür, dass der dazugehörige Speicherbereich freigegeben oder behalten wird, damit dieser beim nächsten Aufruf wieder zurückgeschrieben werden kann. Dabei spielt das Multithreading eine wichtige Rolle, um quasi fließende Übergänge zwischen den Software-Komponenten zu gewährleisten und dass das User Interface auf keinen Fall blockiert wird.

Eine Activity kann über mehrere *Views* verfügen, die jeweils aus unterschiedlichen Textfeldern, Bildern und Buttons bestehen. Sowohl die Views als auch die komplette App kann von Android in den Hintergrund befördert oder auch beendet werden, wenn sie nicht mehr benötigt werden und/oder der Speicher knapp werden sollte. Bei einem erneuten Aufruf der Activity muss sie folglich neu initialisiert werden oder ihr Abbild ist aus dem Speicher wiederherzustellen. Unterstützt werden Apps durch die *Services*, die demgegenüber weitreichendere Rechte besitzen, um mit dem System

und der Hardware zu interagieren. Services ergänzen eine Activity und führen ohne User Interface bestimmte Aufgaben im Hintergrund aus, wie etwa den Aufbau einer Internet-Verbindung bei einer E-Mail-App.

Android generiert bei der Arbeit verschiedene Systemnachrichten (Intents), die den aktuellen Status prozessübergreifend bekannt geben. Der Entwickler kann diese Nachrichten mithilfe von Broadcast-Receivern empfangen und dann situationsbezogen darauf reagieren. Derartige Intents können auch selbst entwickelt und für die App-Steuerung eingesetzt werden.

Bei Android kommen – wie generell bei Linux – verschiedene Dateisysteme parallel zum Einsatz. Für die Speicherung von Anwenderdaten wird traditionell ein spezielles System mit der Bezeichnung YAFS (Yet Another Flash File System) eingesetzt, welches sich insbesondere für Flash-Speicher (NAND) eignet. Außerdem gibt es das *Extended File System* (EXT2, EXT3, EXT4), welches für Linux-Systeme als Standard zu betrachten ist.

Typischerweise sind bei Android-Geräten fünf bis sechs verschiedene Dateisysteme (/proc/filesystems) für die physikalische Speicherung aktiv, wobei es sich bei dem größten Teil (von insgesamt ca. 15–20 unterstützten Systemen) um Dateisysteme handelt, die sich auf *virtuelle Devices* beziehen und die deshalb mit dem Attribut *nodev* versehen sind.

Informationen über die Konfiguration, den Kernel und Prozesse werden mit einem proc-Dateisystem verarbeitet, welches nur einen Root-Zugriff erlaubt, und das insbesondere von Windows her bekannte System FAT32 (VFAT unter Linux/Android) wird für SD-Karten und Onboard-Speicher wie eMMC (Embedded Multimedia Card) eingesetzt. Einige Hersteller, wie beispielsweise Samsung, verwenden zudem auch eigene Dateisysteme wie beispielsweise das *Robust Fat Filesystem* (RFS).

2.2 Daten kopieren

Das Kopieren von Daten eines Android-Gerätes oder auch das Anfertigen von Backups ist eine notwendige Maßnahme, damit wichtige Daten im Fehlerfall oder auch aus Versehen nicht verloren gehen, was insbesondere passieren kann, wenn man eigene Programme erstellt oder auch am Gerät herumbastelt, weshalb hierauf explizit eingegangen wird.

Um Daten zwischen einem Android-Gerät und einem PC auszutauschen, funktioniert dies am besten über eine USB-Verbindung, wofür am Smartphone oder Tablet eine Micro- oder auch Mini-USB-Buchse und am PC eine A-Buchse vorgesehen ist. Näheres zum USB und den Besonderheiten bei Smartphones und Tablets ist in Kapitel 1.4 zu finden.

Die kleinen USB-Stecker sind nicht immer so einfach in die passende Buchse einzuschieben, nicht nur weil sich die Micro- und die Mini-USB-Connectoren so ähnlich sehen, was immer wieder zu Verwechslungen bei den passenden Verbindungskabeln führt, sondern auch weil nicht immer die jeweilige Einsteckrichtung zu erkennen ist. Um nicht durch wiederholtes Probieren die Buchse im Smartphone zu verbeulen, empfiehlt es sich, eine Markierung oben (oder unten) am Stecker anzubringen, was sich bei schwarzen Kabeln mit einem Klecks Tipp-Ex leicht bewerkstelligen lässt. Die Buchsen in den Geräten sind keineswegs mit der gleichen Orientierung eingebaut, auch wenn es sich um den gleichen Typ handeln sollte.

Die Android-Geräte werden vom PC standardmäßig als Wechseldatenträger, also als USB-Sticks erkannt, sodass die Daten auf die übliche Art und Weise ausgetauscht werden können. Bei einem aktuellen Windows-PC ist (zunächst) keine besondere Software notwendig, höchstens bei Windows XP, welches noch nicht mit derartigen Datenträgern automatisch umgehen kann, sodass hierfür ein spezieller Treiber vom Hersteller des Android-Gerätes benötigt wird.

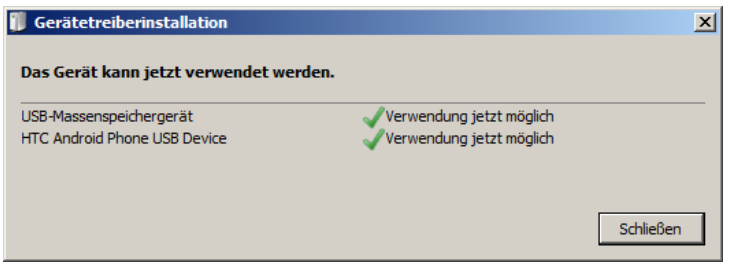

Abb. 2–3 Der Windows-PC hat das angeschlossene Smartphone erkannt.

Das Android-Gerät kann beim Erkennen einer USB-Verbindung mit einem PC nach dem zu verwendenden Verbindungstyp fragen, wofür es typischerweise die Optionen *Nur laden*, *Sync.*, *Festplatte*, *USB-Verbindung* (mobiles Netzwerk für PC freigeben) und *Internet-Durchgang* (über den PC mit dem Internet verbinden) gibt.

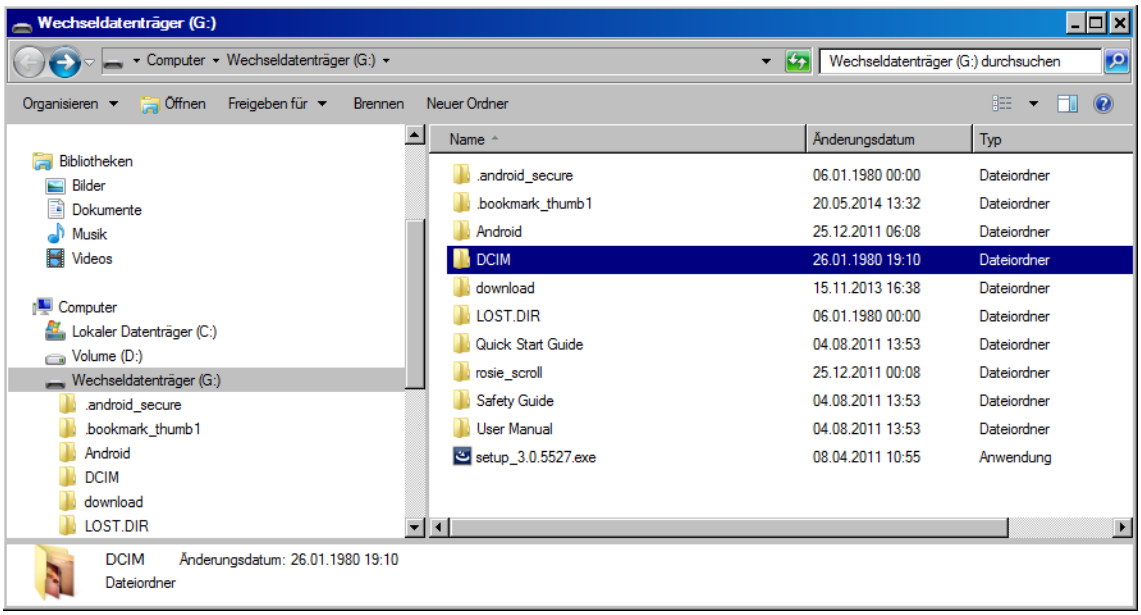

Abb. 2–4 Das Android-Smartphone wird von Windows als üblicher Wechseldatenträger angesehen, sodass das Hin- und Herkopieren von Daten, wie etwa von Fotos im DCIM-Verzeichnis, ohne Weiteres möglich ist.

Falls für das Android-Gerät ein Betriebssystemtreiber oder eine spezielle Software vorhanden ist, lassen sich die Daten auf den Geräten miteinander synchronisieren und spezielle Funktionen, wie etwa das Neuschreiben der Firmware und Ähnliches, nutzen. Im Grunde genommen hat jeder Hersteller hierfür eine Lösung parat, beispielsweise bietet HTC den *HTC Sync Manager* und Samsung das Programm *Kies*, welches eine der umfassenderen Lösungen darstellt.

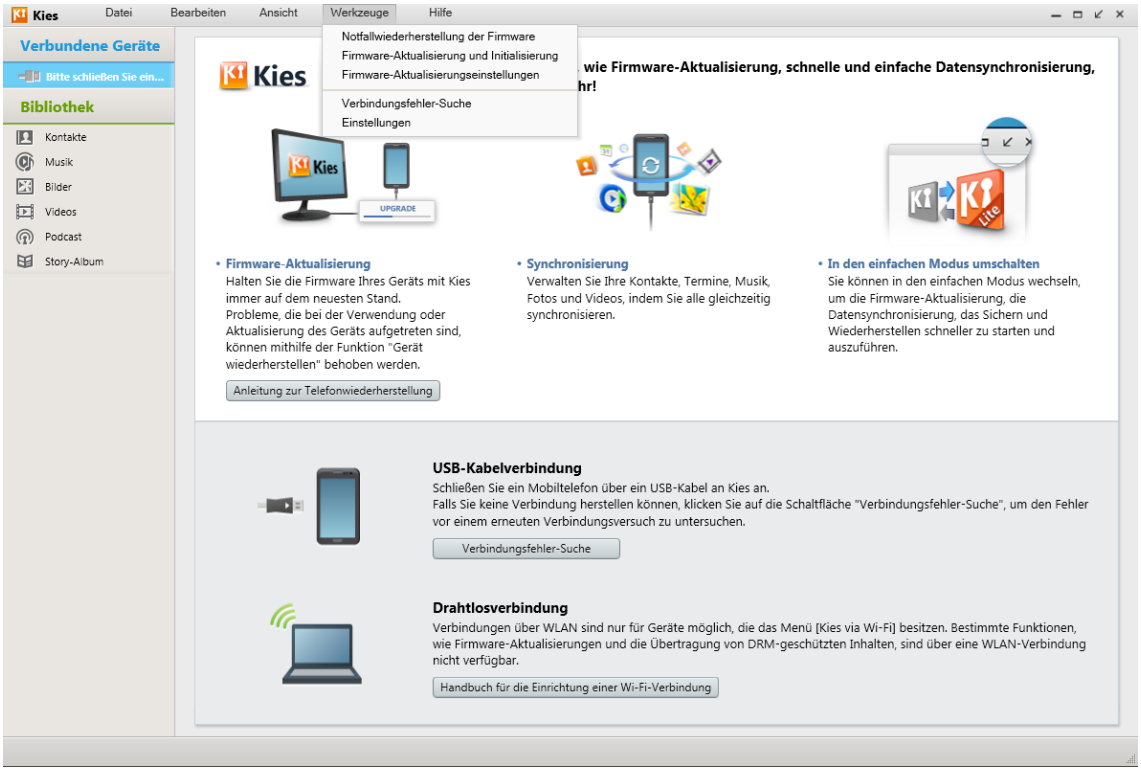

Abb. 2–5 Das Programm Kies für Geräte der Firma Samsung bietet eine Reihe nützlicher Funktionen.

Für das Kopieren – oder auch Verschieben und Löschen – von Daten innerhalb eines Systems kommt üblicherweise ein Dateimanager zum Einsatz, der auch eine Orientierung in der Verzeichnisstruktur bietet. Android liefert einen geeigneten Dateimanager standardmäßig jedoch nicht mit. Für PCs ist der Total Commander einer der bekanntesten und leistungsfähigsten Dateimanager mit zahlreichen Erweiterungen wie integriertem Editor, Packer und FTP-Client, der in einer angepassten Version auch für Android verfügbar ist.

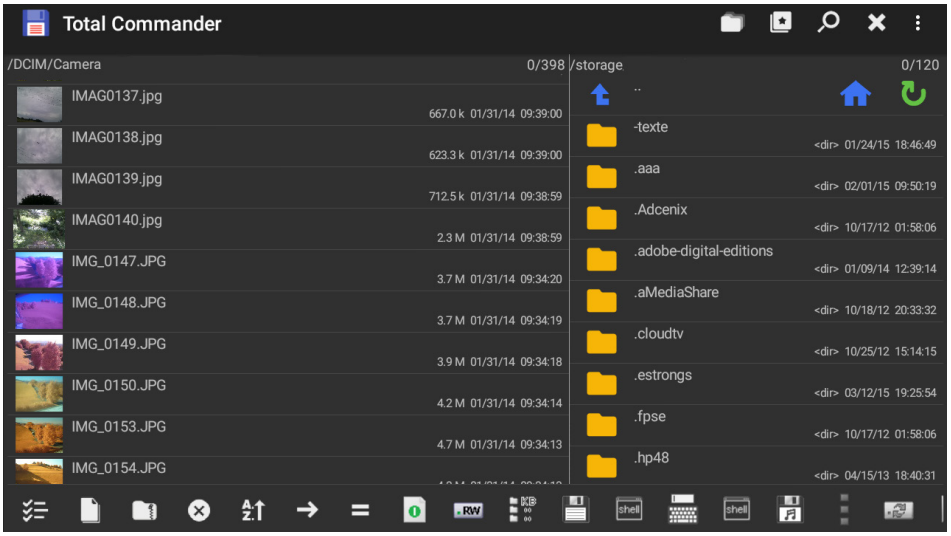

Abb. 2–6 Der bereits aus alten Windows-Zeiten bekannte Total Commander ist auch für Android verfügbar und bietet die praktische Zweispaltenansicht, etwa um Daten auf die SD-Card zu verschieben.

2.3 Entwicklermodus – USB-Debugging

Für bestimmte Arbeiten ist bei Android der Entwicklermodus zu aktivieren, etwa um mit dem Gerät in Android Studio zu kommunizieren, den Bootloader nutzen oder auch um Custom-ROMs aufspielen zu können. Üblicherweise firmiert dieser Modus unter *Debugging* oder *USB-Debugging*, weil das Android-Gerät hiermit über den USB mit einem Computer für den Austausch von systemrelevanten Daten kommunizieren kann.

Die Option *USB-Debugging* ist je nach Version und Modell an verschiedenen Stellen zu erreichen, etwa über Apps – Einstellungen – Anwendungen – Entwicklung oder auch über Einstellungen – System – Entwickleroptionen.

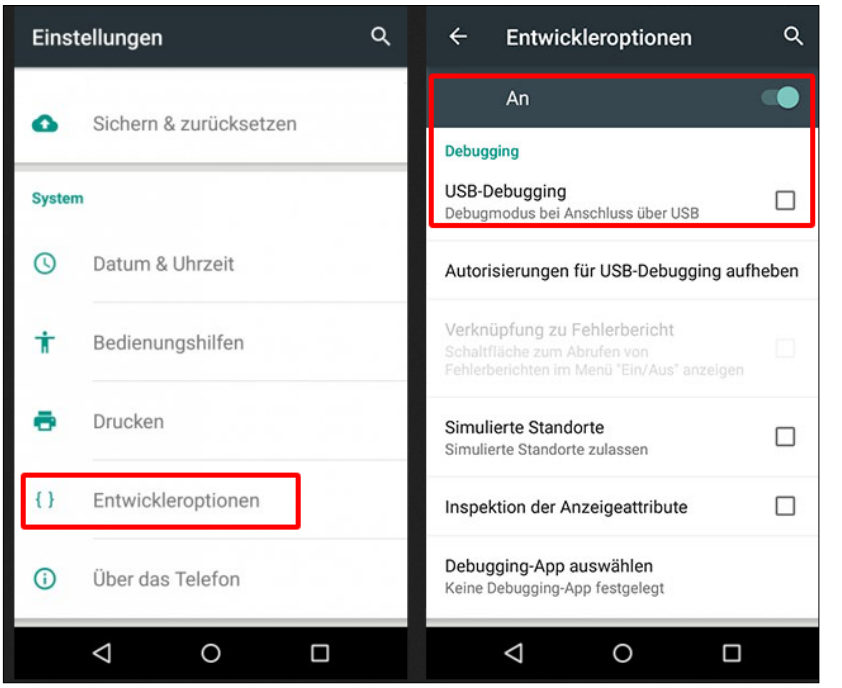

Abb. 2–7 Einschalten von USB-Debugging

Falls sich für die Aktivierung der Entwickleroptionen kein Menüpunkt finden lassen sollte, funktioniert meist der folgende Trick: Bei den Einstellungen auf *Über das Telefon* tippen, ganz nach unten scrollen und kurz (mindestens) siebenmal auf die Zeile *Buid-Nummer* tippen, bis die Meldung *Entwickler* (oder okay, Sie sind bereits Entwickler) erscheint, woraufhin die verschiedenen Entwickleroptionen zur Verfügung stehen.

2.4 Backup

Neben dem direkten Kopieren von Daten auf einen PC, was als einfachste und zuverlässigste Methode für die Datensicherung bei einem Android-Gerät gilt, gibt es auch spezielle Backup-Optionen, wofür möglicherweise bereits standardmäßig eine entsprechende App auf dem jeweiligen Gerät installiert ist. Fast jeder Smartphone-Hersteller betreibt eigene Backup-Lösungen wie etwa Sony, LG, HTC und Samsung. Was dabei im Einzelnen gesichert wird, ist allerdings nicht immer zweifelsfrei zu erkennen.

Als Speichermedium ist zunächst die SD-Card geeignet, weil hier die Daten – im Gegensatz zu den Daten im internen Gerätespeicher – auch nach einem Rücksetzen auf die Werkseinstellung erhalten bleiben. Ab Android 4.0 gibt es standardmäßig in

den Einstellungen den Punkt Speicher – Verschiedenes, womit sich Multimediadaten (Musik, Fotos, Videos) auf die SD-Karte oder per USB-Kabel direkt auf einen PC übertragen lassen.

Ein Nachteil der herstellerspezifischen Lösungen ist, dass sich die Daten nicht auf Geräte eines anderen Herstellers übertragen lassen, weshalb es eine ganze Reihe von geräteunabhängigen Backup-Lösungen mit einem unterschiedlichen Leistungsumfang und für verschiedene Android-Versionen gibt. Apps wie *MyBackup* (ab Android 1.5), *Super Backup* (ab Android 2.01) oder *Helium Desktop* (ab Android 4.0) erlauben recht genaue Einstellungen, was (Fotos, Musik, Videos, Kontakte, Anruflisten, Lesezeichen, SMS, Kalendereinträge, Einstellungen Homescreen, Wecker, Wörterbücher, Musik-Playlisten, Apps, Daten) im Einzelnen wo (SD-Card, Computer, Cloud) gesichert werden soll.

In Abhängigkeit von den zu sichernden Daten – beispielsweise für Einstellungen und System-Apps – sind Root-Rechte (siehe Kapitel 2.5) für das Kopieren bzw. Sichern notwendig, wie etwa bei *Titanium Backup Root* oder *Clockwork Mod Recovery,* welches zudem ein vollständiges System-Image sichern kann. Das Tool *Holo Backup* gestattet ebenfalls eine Image-Sicherung, allerdings ohne System-Apps, dafür benötigt es auch keine Root-Rechte und funktioniert ohne Installation unter Windows und Linux.

Abb. 2–8 Titanium Backup ist eine umfassende Backup-Lösung, die jedoch Root-Rechte (Superuser) erfordert.

Das Speichern von Daten bei einem Cloud-Dienst (Dropbox, SkyDrive, Google Drive) ist ebenfalls eine gängige Maßnahme, auch wenn der Datenschutz – nach deutschen bzw. europäischen Aspekten – dabei keineswegs sichergestellt ist, sodass hier keine Passwörter und private Daten abgelegt werden sollten, weil die Übertragung oftmals sogar unverschlüsselt erfolgt.

2.5 Geräte rooten

Bei Windows gibt es den *Administrator* und bei Linux den *Superuser* oder *Root*, denen ein voller Systemzugriff gestattet ist. Weil Android von Linux abstammt, gibt es auch hier einen Nutzer *Root*, der aber standardmäßig deaktiviert ist, weil das Arbeiten des Benutzers mit Root-Rechten aus Sicherheitsgründen nicht vorgesehen ist. Android verfügt – im Gegensatz zu den anderen Systemen – jedoch nicht über eine Benutzerverwaltung. Stattdessen wird für jede installierte Anwendung ein eigener Benutzer angelegt, wobei die Anwendung eine bestimmte User ID führt und die jeweiligen Rechte hierfür festlegt, sodass die Root-Aktivierung bei Android eigentlich die Lese- und Schreibrechte auf die Apps und Services verändert.

Dem Vorteil, nahezu alle möglichen Systemmanipulationen inklusive kompletter Backups als Root oder Superuser vornehmen zu können, wie etwa auch vorinstallierte Apps oder Werbe-Apps (Bloatware) löschen zu können oder auch Funktionen nachzurüsten, steht zunächst der Nachteil gegenüber, dass bei einem »gerooteten« Smartphone die Gerätegarantie erlischt. Die Gewährleistung bleibt allerdings bestehen, denn sie ist eine gesetzliche Pflicht, die der Verkäufer einhalten muss, wenn der Kunde einen Mangel am Gerät feststellt, der schon zum Zeitpunkt des Kaufs bestand.

Die Garantie ist eine freiwillige Leistung des Herstellers oder Verkäufers, die dem Kunden die Funktionsfähigkeit der Ware über einen bestimmten Zeitraum garantiert. Stellt dieser einen Defekt fest, der bereits zum Zeitpunkt des Kaufs bestand, greift daher die Garantie, die von vielen Herstellern bei einem »gerooteten« Gerät verweigert wird. Die Hersteller verfahren in solchen Fällen jedoch nicht einheitlich. HTC gibt beispielsweise an, dass die Garantie bestehen bleibt, wenn der Defekt nicht durch das Rooten entstanden ist. Das Rooten ist zwar nicht komplett rückgängig zu machen – auch wenn Programme wie Framaroot oder Kingo Root eine Unroot-Funktion zur Verfügung stellen –, gleichwohl empfiehlt es sich, bei Garantieansprüchen so viel wie möglich von dem, was man mit Root bewerkstelligt hat, wieder zu beseitigen.

Wenn man als Root unterwegs ist, muss man sich über die Gefahren, die damit einhergehen können, klar sein, denn nicht nur unbeabsichtigte Aktionen können das Gerät »lahmlegen«, sondern bestimmte Schadsoftware kann einen höheren Schaden verursachen, als wenn sie nur auf eine übliche Benutzerumgebung treffen würde.

Die Ausführung eines Root-Vorganges kann auch schiefgehen, wobei man hier zwischen einem Soft-Brick und einem Hard-Brick unterscheidet. Die Bezeichnung *Brick* soll verdeutlichen, dass das Gerät bei einem derartigen Fehler nur noch den Nutzen eines Ziegelsteins hat. Ein Soft-Brick ist aber lediglich ein Software-Fehler, der dazu führt, dass das Gerät nicht mehr richtig startet, was sich meist durch ein Rücksetzen auf die Werkseinstellungen oder die Installation einer neuen Firmware beheben lässt. Seltener ist ein Hard-Brick, bei dem das System so weit zerstört wurde, dass überhaupt kein Zugriff mehr auf das Gerät möglich ist und auch keine Rettungsmaßnahme mehr greifen kann.

Zur Erlangung von Root-Rechten unter Android gibt es keine einheitliche Methode. Grundsätzlich kann das Rooten mit einer App oder mit einer speziellen PC-Software erfolgen, wofür dann zwischen PC und Android-Gerät eine USB-Verbindung hergestellt werden muss. Außerdem ist der USB-Debugging-Modus beim Android-Gerät hierfür zu aktivieren (Entwickler-Modus, siehe 2.3). Dies sind auch die Voraussetzungen, um eine spezielle geroutete Firmware auf Android-Geräte übertragen zu können (siehe 2.6).

Vor dem Beginn irgendeiner Rooting-Prozedur sollte klar sein, aus welchem Grunde man die Root-Rechte benötigt und was man damit anfangen will, denn sowohl für den Backup als auch für das Entfernen von Bloatware existieren zum einen Lösungen, die Root-Rechte benötigen, zum anderen aber auch solche, die eben keine voraussetzen. Wer bestimmte Systemfunktionen steuern möchte, etwa um den Prozessor für einen niedrigeren Stromverbrauch (Undervolting u. Ä.) oder eine höhere Taktrate einzustellen, kommt um die Root-Funktionalität allerdings nicht herum.

Im Internet gibt es eine Vielzahl von Anleitungen für das Rooten von Android-Geräten, etwa auf *http://www.android-hilfe.de/* oder *http://www.xda-developers.com/* oder *http://www.androidcentral.com/root*, die sich generell für die Beschaffung von Informationen rund um Android eignen und auch zahlreiche Foren bieten.

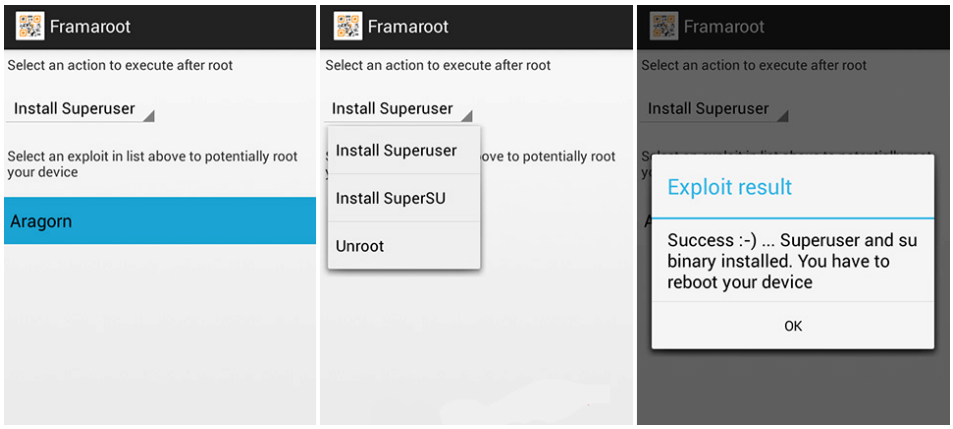

Abb. 2–9 Framaroot hat das Gerät erfolgreich »gerootet« und den Superuser erstellt.

Einige Rooting-Programme wie etwa *Framaroot* funktionieren mit unterschiedlichen Android-Geräten wie eine übliche App, sodass der Vorgang dann relativ einfach durchzuführen ist. Die Geräteunterstützung bei den Rooting-Apps fällt recht unterschiedlich aus; während *Framaroot* regelmäßig aktualisiert wird und somit eine breite aktuelle Gerätepalette unterstützt, sind Tools wie *Root Genius* vorwiegend für Geräte aus China (HTC, Huawei) geeignet und auch nur in chinesischer Sprache verfügbar (was für den Rooting-Vorgang allerdings kaum relevant ist). Rooting-Apps wie *Universal Androot* oder *Towelroot* unterstützen hingegen nur ältere Android-Geräte. Die Rooting-Apps nutzen bekannte Sicherheitslücken in Android zum Rooten aus. Bei Framaroot (Abbildung 2–9, Aragon) werden diese nach Figuren aus »Der Herr der Ringe« benannt.

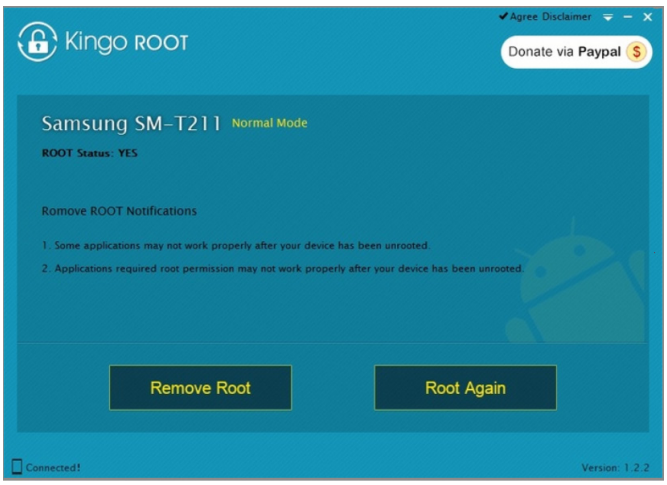

Abb. 2–10 Kingo ROOT kann das Gerät auch wieder in den normalen (ungerooteten) Zustand versetzen.

Kingo Root (Abbildung 2–10) ist ein Windows-Programm, welches den Root-Vorgang ebenfalls für verschiedene Android-Geräte (mit nur einem Mausklick) ausführen kann. Wie üblich muss hierfür die Option *USB-Debugging bei dem Android-Gerät aktiviert sein. Eine illustrierte Anleitung für das Programm macht es auch Einsteigern leicht, die Root-Funktionalität herzustellen. Ähnlich funktioniert das Programm UnlockRoot (Pro), wofür die Android ADB Drivers* des Android SDK zu installieren sind (Näheres hierzu in Kapitel 2.8.1).

Ob man Root-Rechte erlangt hat, stellt man am besten dadurch fest, dass eine App, die tatsächlich Root-Rechte benötigt, aufgerufen wird, oder man probiert die App *Root Checker*, die explizit die Root-Funktionalität detektieren kann.

Abb. 2–11 Die Superuser-Anfrage taucht bei einem »gerooteten« Gerät automatisch auf.

Üblicherweise wird beim Root-Vorgang eine App wie *SuperSU* oder *Superuser* mitinstalliert, die als Steuerungsinstanz fungiert, ähnlich wie die Benutzerkontensteuerung bei Windows. Wenn eine Anwendung Root-Rechte anfordert, wird dann ein Warnhinweis (Abbildung 2–10) mit einer Anfrage angezeigt, sodass die Einwilligung vom Benutzer an dieser Stelle erlaubt oder nicht gestattet werden kann, was somit keine großen Risiken birgt.

2.6 Alternative Firmware – Custom ROMs

Laufend erscheinen neue Android-Versionen, was meist mit der Vorstellung neuer Geräte einhergeht, für die die Hersteller eine begrenzte Zeit lang noch Updates für kommende Android-Versionen zur Verfügung stellen. Nicht selten erscheint ein Smartphone bereits nach kürzester Zeit als »altes Eisen«, weil die aktuellen Modelle