Erste Schritte mit Android

Mit dem Appium-Plugin lassen sich Tests für Apps auf Android- oder iOS-Mobilgeräten erstellen und ausführen. Dieses Tutorial beschreibt das grundsätzliche Vorgehen anhand eines mitgelieferten Beispiels für Android, bestehend aus einer einfachen App und einer expecco-Testsuite.

Die App *expecco Mobile Demo* berechnet und überprüft verschiedene alltägliche Codes: die IBAN aus dem europäischen Zahlungsverkehr, die internationalen GTIN-13-Produktcodes, wie man sie bei Strichcodes im Einzelhandel findet, und die Seriennummern auf Euro-Banknoten.

Die Testsuite enthält Testfälle für einzelne Funktionen der App. Dabei sind noch nicht alle Funktionen abgedeckt, sondern werden im Laufe des Tutorials ergänzt.

Schritt 1: Installation und Konfiguration

Für dieses Tutorial benötigen Sie eine Installation von expecco inkl. Plugins sowie das Mobile Testing Supplement. Laden Sie dazu die Installationsprogramme für expecco, für die expecco Plugins sowie für das Mobile Testing Supplement herunter und installieren Sie die Programme in dieser Reihenfolge. Folgen Sie dabei den Anweisungen des jeweiligen Installers. Wir empfehlen Ihnen bei der Installation des Mobile Testing Supplements den Universal-ADB-Treiber mit zu installieren wenn Sie danach gefragt werden. Dieser vereint Treiber für ein breites Spektrum an Android-Geräten, sodass Sie nicht für jedes Gerät einen eigenen Treiber suchen und installieren müssen.

Außerdem benötigen Sie für die Ausführung ein Android-Gerät, das Sie über USB mit dem Rechner verbinden können, auf dem expecco installiert ist, oder einen Emulator. Optional können Sie noch ein zweites Gerät benutzen.

Starten Sie nun expecco und installieren Sie eine geeignete Lizenz.

Bevor Sie loslegen sollten Sie die Einstellungen des Mobile Testing Plugins überprüfen und ggf. anpassen. Öffnen Sie im Menü den Punkt *Extras* > *Einstellungen* und dort unter *Erweiterungen* den Eintrag *Mobile Testing* (Abb. 1) Zu Beginn ist für sämtliche Pfade eingestellt, dass expecco diese automatisch setzt (1). Um einen anderen Pfad einzutragen, schalten Sie diese Option ab.

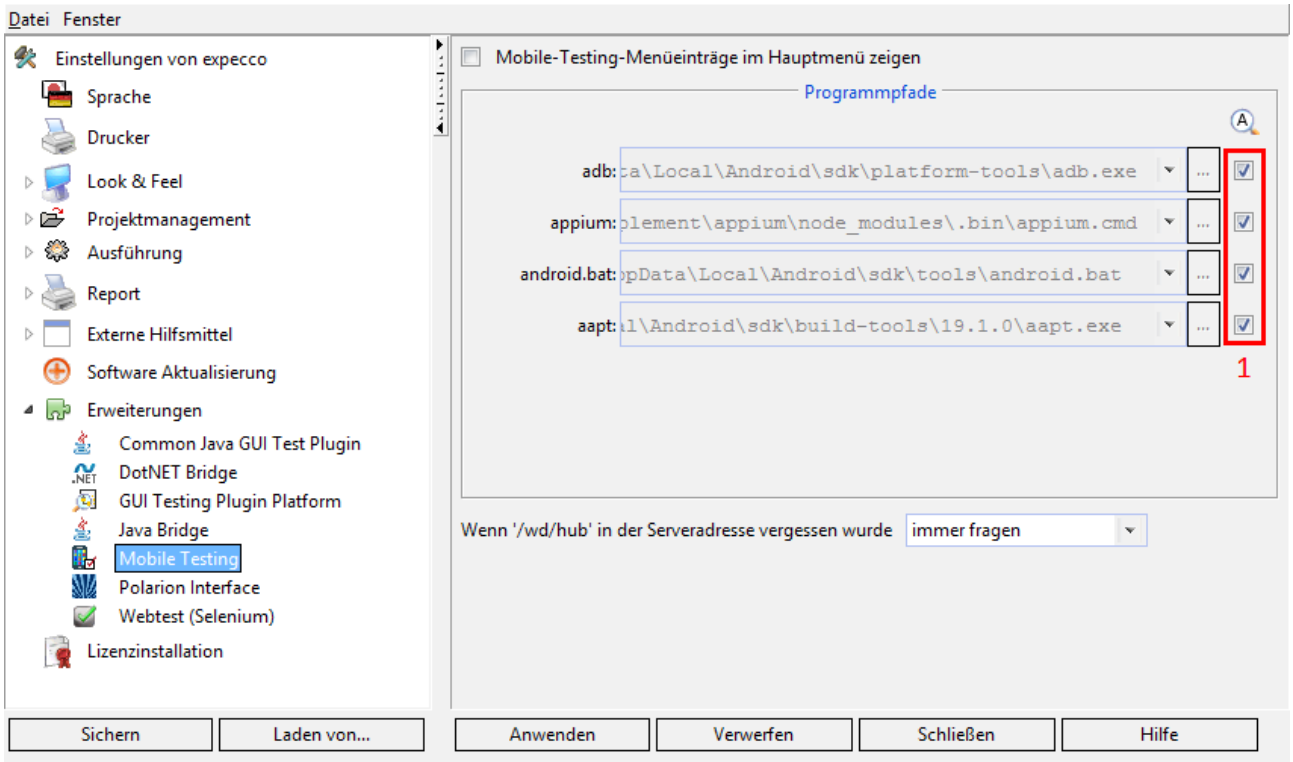

Abb. 1: Konfiguration des Plugins

Stellen Sie für die Pfade zu *adb*, *appium* und *aapt* sicher, dass sie richtig angegeben sind. Wird ein Pfad als ungültig erkannt, wird das entsprechende Eingabefeld rot markiert. Die Datei *android.bat* wird nur zum Starten des *SDK Managers* und des *AVD Managers* benötigt. Da diese für dieses Tutorial nicht relevant sind, können Sie diesen Pfad ignorieren. Übernehmen Sie die Änderungen.

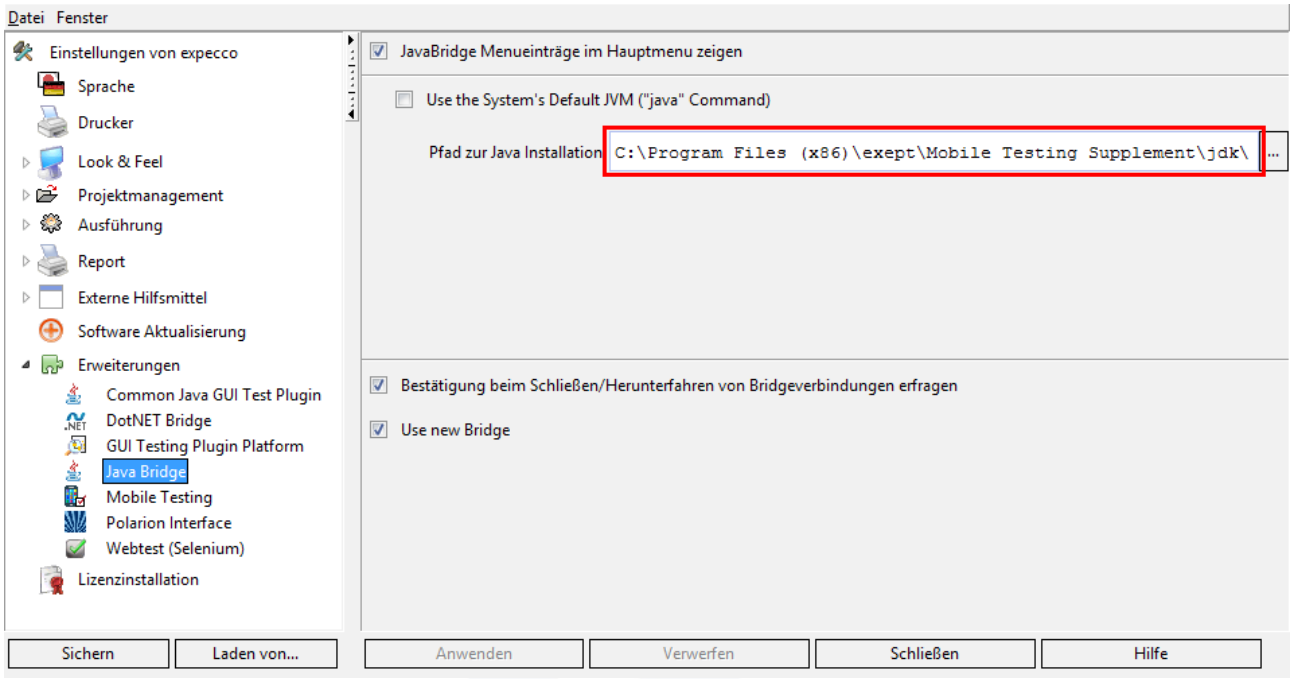

Abb. 2: Konfiguration des JDKs

Wechseln Sie als nächstes zum Eintrag *Java Bridge* (Abb. 2). Hier muss der Pfad zu Ihrer Java-Installation angegeben werden. Tragen Sie hier ein JDK ein. Falls Sie das aus dem Mobile Testing Supplement verwenden möchten, lautet der Pfad

C:\Program Files (x86)\exept\Mobile Testing Supplement\jdk

Übernehmen Sie die Änderung und klicken Sie anschließend auf Sichern, um alle Einstellungen dauerhaft zu speichern. Schließen Sie anschließend den Einstellungsdialog.

Schritt 2: Demo ausführen

Nun ist das Plugin soweit konfiguriert, dass Sie damit Tests erstellen können. In diesem Tutorial wollen wir die Funktionen der Android-App *expeccoMobileDemo* testen. Als Grundlage dafür dient die Testsuite *m02_expeccoMobileDemo.ets*, die Sie in den Beispielen zu expecco finden. In dieser befindet sich bereits ein vorgefertigter Testplan mit einigen Testfällen für diese App. Öffnen Sie die Testsuite über die Schaltfläche *Beispiel aus Datei* (Abb. 3).

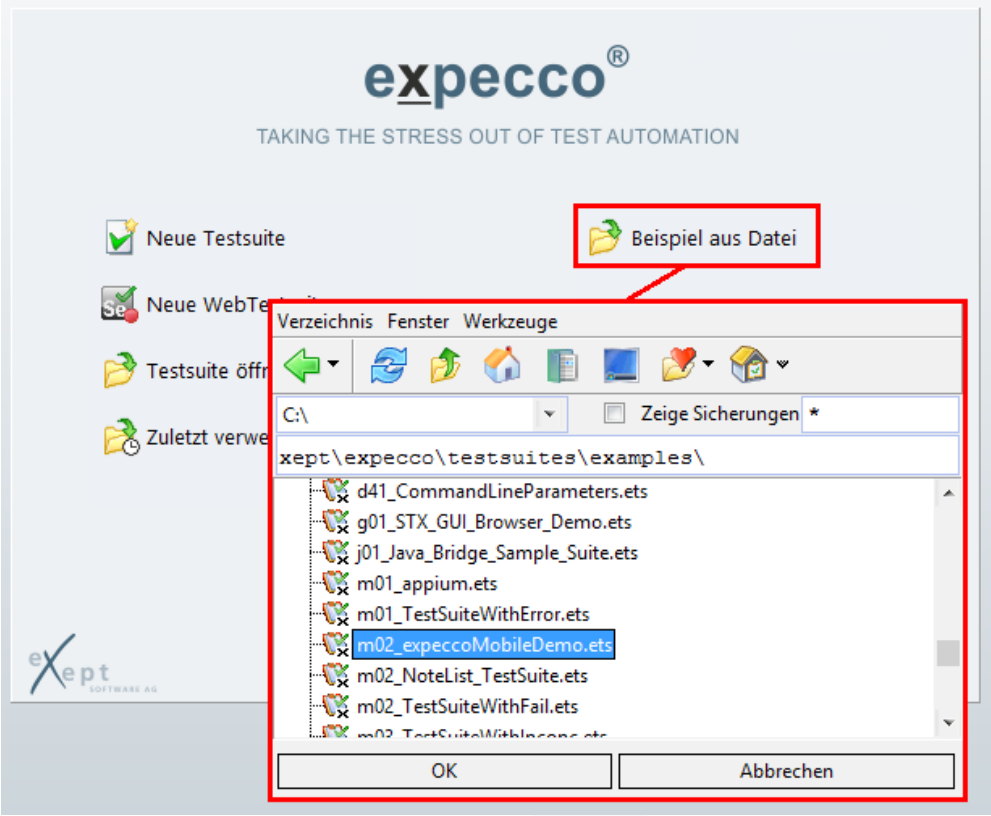

Abb. 3: Beispiel-Testsuite öffnen

In der Testsuite ist das Paket der Demo-App als Anhang enthalten (*expeccoMobileDemo-debug.apk*). Mithilfe des bereitgestellten Bausteins *Export Demo App* können Sie die Datei an einen beliebigen Ort auf Ihrem Rechner exportieren. Wählen Sie dazu den Baustein aus (1) und klicken Sie auf den grünen Play-Knopf (2) um den Baustein auszuführen (Abb. 4). Der Baustein öffnet einen Dateidialog, in dem Sie angeben, wo das Paket gespeichert werden soll.

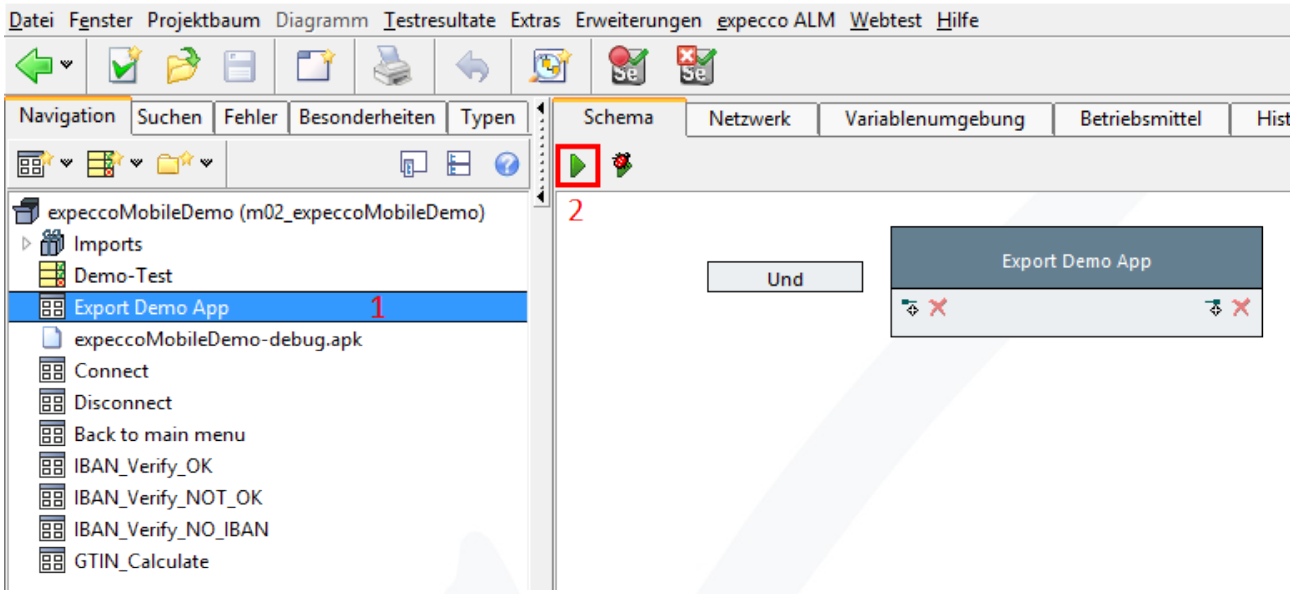

Abb. 4: App exportieren

Bevor wir uns mit dem weiteren Inhalt der Testsuite beschäftigen, konfigurieren Sie zuerst die Verbindung und welches Gerät Sie benutzen wollen. Schließen Sie dazu ein Gerät über USB an Ihren Rechner an oder starten Sie einen Emulator.

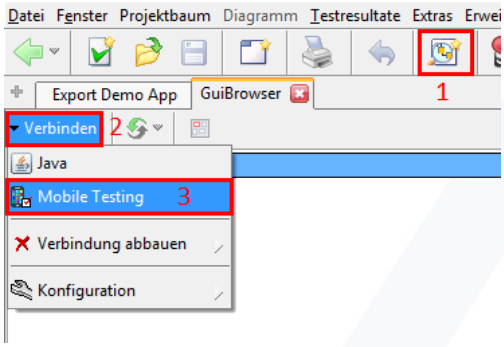

Abb. 5: Verbindungseditor öffnen

Öffnen Sie nun den GUI-Browser (1) und wählen Sie unter *Verbinden* (2) den Eintrag *Mobile Testing* (3) (Abb. 5), um den Verbindungsdialog zu öffnen.

Sie sehen eine Liste aller angeschlossenen Android-Geräte (1) (Abb. 6). Sollte Ihr Gerät nicht in der Liste auftauchen, stellen Sie sicher, dass es eingeschaltet und über USB verbunden ist. Außerdem müssen Sie auf dem Gerät in den Einstellungen unter *Entwickleroptionen* USB-Debugging einschalten. Falls Sie die Entwickleroptionen in den Einstellungen Ihres Gerätes nicht finden, müssen Sie diese zuerst aktivieren, indem Sie in den Einstellungen den Menü-Eintrag *Über das Telefon* öffnen und dort siebenmal auf *Build-Nummer* tippen. Außerdem müssen Sie auf dem Gerät USB-Debugging von Ihren Computer zulassen. Falls das noch nicht aktiviert ist und auf dem Gerät kein entsprechender Dialog erscheint, kann es helfen, das Gerät aus- und wieder einzustecken. Das kann insbesondere dann passieren, wenn Sie den ADB-Treiber installiert haben während das Gerät bereits über USB angeschlossen war.

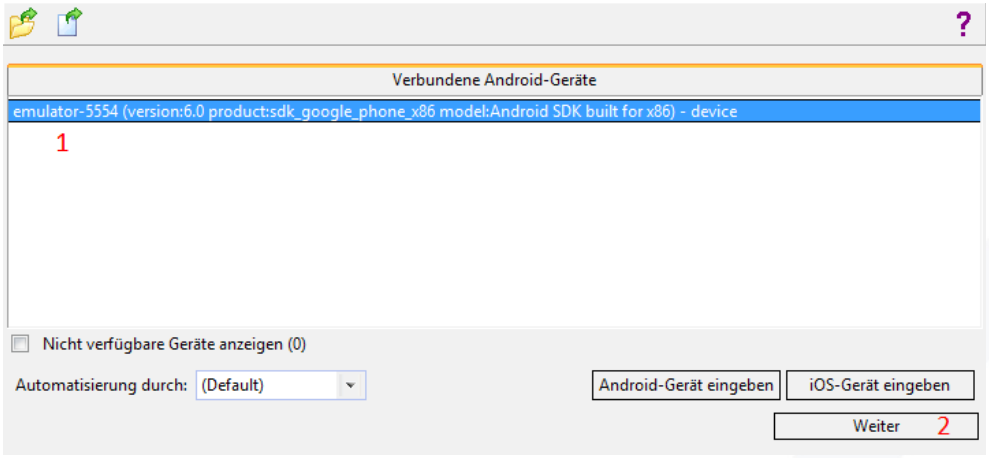

Abb. 6: Gerät im Verbindungsdialog auswählen

Haben Sie Ihr Gerät in der Liste gefunden, wählen Sie es aus und klicken Sie auf *Weiter* (2).

Als nächstes geben Sie an, welche App Sie verwenden wollen (Abb. 7). Dabei können Sie wählen, ob Sie eine App starten möchten, die bereits auf dem Gerät installiert ist (*App auf dem Gerät*) oder ob eine App installiert und gestartet werden soll (*App installieren*). Für den Fall, dass Sie eine bereits installierte App benutzen wollen, erhalten Sie eine Liste aller auf dem Gerät installierten Pakete (1), die in Systempakete und Fremdpakete (2) unterteilt sind, sowie deren Activities (3). Diese können Sie dann einfach in den jeweiligen Feldern auswählen.

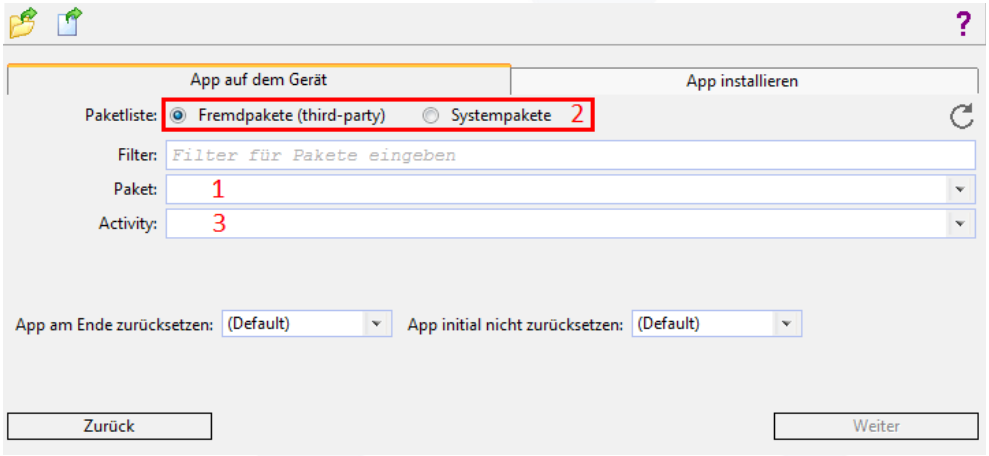

Abb. 7: Auf dem Gerät installierte App angeben

Für dieses Tutorial soll die App installiert werden, die Sie eben aus der Testsuite exportiert haben. Wählen Sie also *App installieren* aus und tragen Sie bei *App* (1) den entsprechenden Pfad ein (Abb. 8). Sie können den Knopf links benutzen (2), um einen Dateidialog zu öffnen, mit dem Sie zu der Datei navigieren können, um sie einzugeben. Das Paket (3) und die Activity (4) der App werden automatisch eingetragen. Sollte die App mehrere Activities besitzen, können Sie die gewünschte auswählen. Klicken Sie nun auf *Weiter* (5).

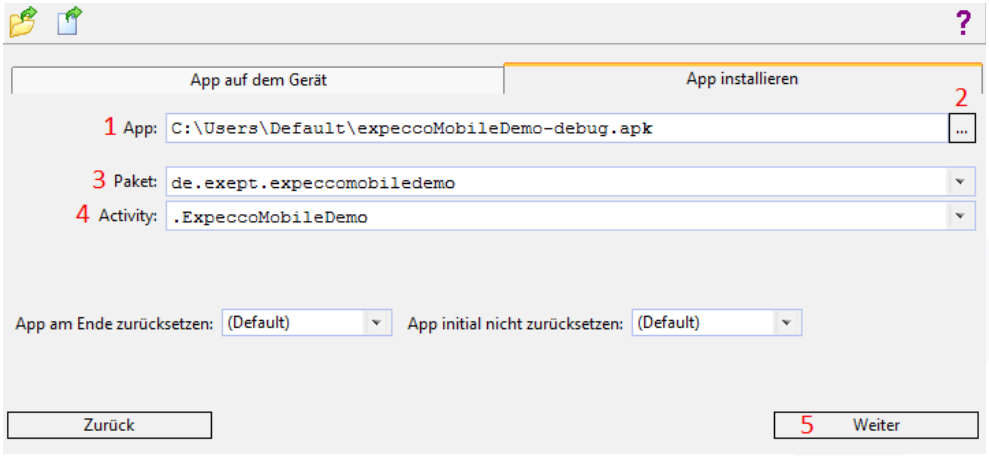

Abb. 8: App angeben, die auf dem Gerät installiert werden soll

Auf der letzten Seite sehen Sie eine Übersicht aller bisherigen Angaben (1) (Abb. 9). Darunter können Sie einen Namen für die Verbindung angeben, unter dem sie im GUI-Browser angezeigt wird (2). Außerdem lässt sich eine Verbindung über diesen Namen identifizieren und in Bausteinen verwenden; der Name muss daher eindeutig sein. Falls Sie keinen Namen angeben, wird generisch einer erzeugt. Geben Sie als Namen *expeccoMobileDemo* ein. Im Feld darunter ist die Adresse zum Appium-Server eingetragen (3). Appium ist die Schnittstelle, über die die angeschlossenen Geräte gesteuert werden. Für dieses Tutorial wird die Verwaltung der Instanzen des Appium-Servers von expecco übernommen. Dafür ist die lokale Standard-Adresse *http://localhost:4723/wd/hub* eingetragen und die Option *Bei Bedarf starten* aktiviert (4). expecco erkennt dann automatisch, ob an der Adresse ein Appium-Server läuft und startet und beendet ihn nötigenfalls automatisch. Wenn der Port *4723* bereits belegt ist oder wenn Sie einmal mehrere Verbindungen parallel betreiben wollen, verwenden Sie an dieser Stelle entsprechend einen anderen Port. Es ist dabei üblich die ungeraden Portnummern oberhalb von *4723* zu verwenden, also *4725*, *4727* usw. Natürlich können Sie auch entfernte Server verwenden, das automatische Starten und Beenden eines Servers kann expecco aber nur lokal für Sie übernehmen.

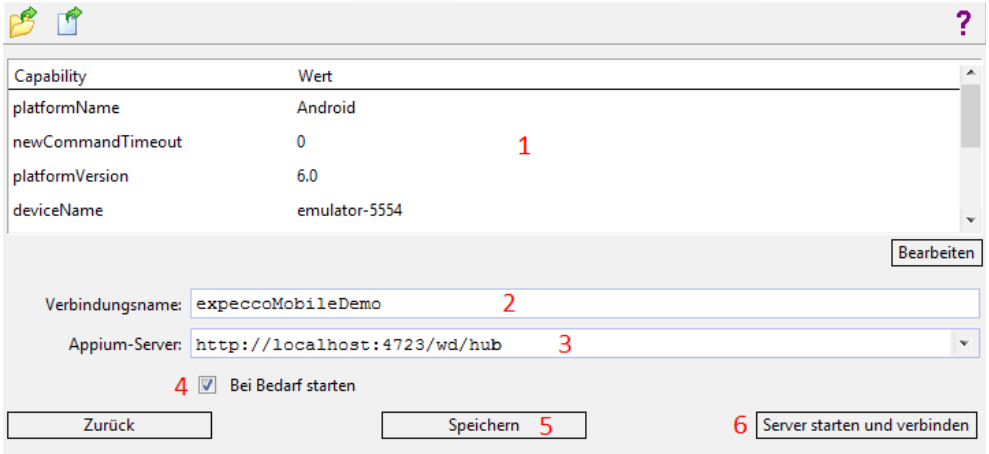

Abb. 9: Verbindungsnamen und Appium-Server konfigurieren

Klicken Sie nun auf *Speichern* (5) um die Einstellungen für die Testausführung zu speichern. Einstellungen können als Anhang einer Testsuite oder in eine externe Dateie gespeichert werden (Abb. 10). Falls Sie mehrere Projekte gleichzeitig offen haben, können Sie in der Liste das Projekt auswählen, in dem der Anhang angelegt werden soll. Klicken Sie auf *Speichern* im Bereich *Einstellungen im Anhang speichern* und geben Sie als Name *expeccoMobileDemo* an. Klicken Sie nun auf *Server starten und verbinden* (6) um mit der angegebenen Konfiguration eine Verbindung herzustellen.

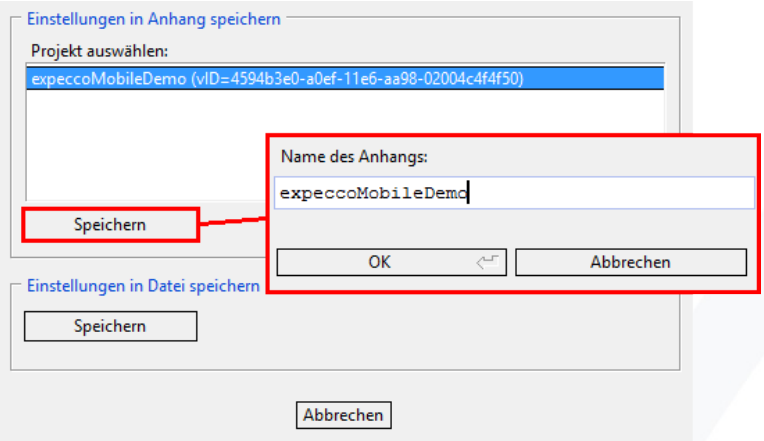

Abb. 10: Einstellungen speichern

Der Verbindungsaufbau kann eine Weile dauern. Warten Sie bis die Verbindung aufgebaut ist und im GUI-Browser angezeigt wird. Sie sehen, dass die App auf dem Gerät gestartet wird. Nun wissen Sie, dass die Konfiguration funktioniert. Die gespeicherten Einstellungen sollen nun für den Test verwendet werden, der dann die gleiche Verbindung aufbaut. Wählen Sie die Verbindung im GUI-Browser aus, machen Sie einen Rechtsklick und wählen Sie im Kontextmenü *Verbindung abbauen*, damit es zu keinem Konflikt kommt. Wechseln Sie dann zurück zum Reiter der Testsuite.

In der Testsuite wurden die Einstellungen als Anhang *expeccoMobileDemo* angelegt (Abb 11). Wählen Sie den Baustein *Connect* (1) aus und wechseln Sie rechts zur Ansicht *Netzwerk* (2). Ziehen Sie per Drag-and-drop die Einstellungen in das Netzwerk des Bausteins (3). Verbinden Sie den Ausgangspin *pathName* mit dem Eingangspin *stringOrFilename[1]* des Bausteins *Connect from File* (4). Mit *Übernehmen* (5) bestätigen Sie die Änderungen. Dieser Baustein wird zu Beginn des Tests die Verbindung zur App herstellen.

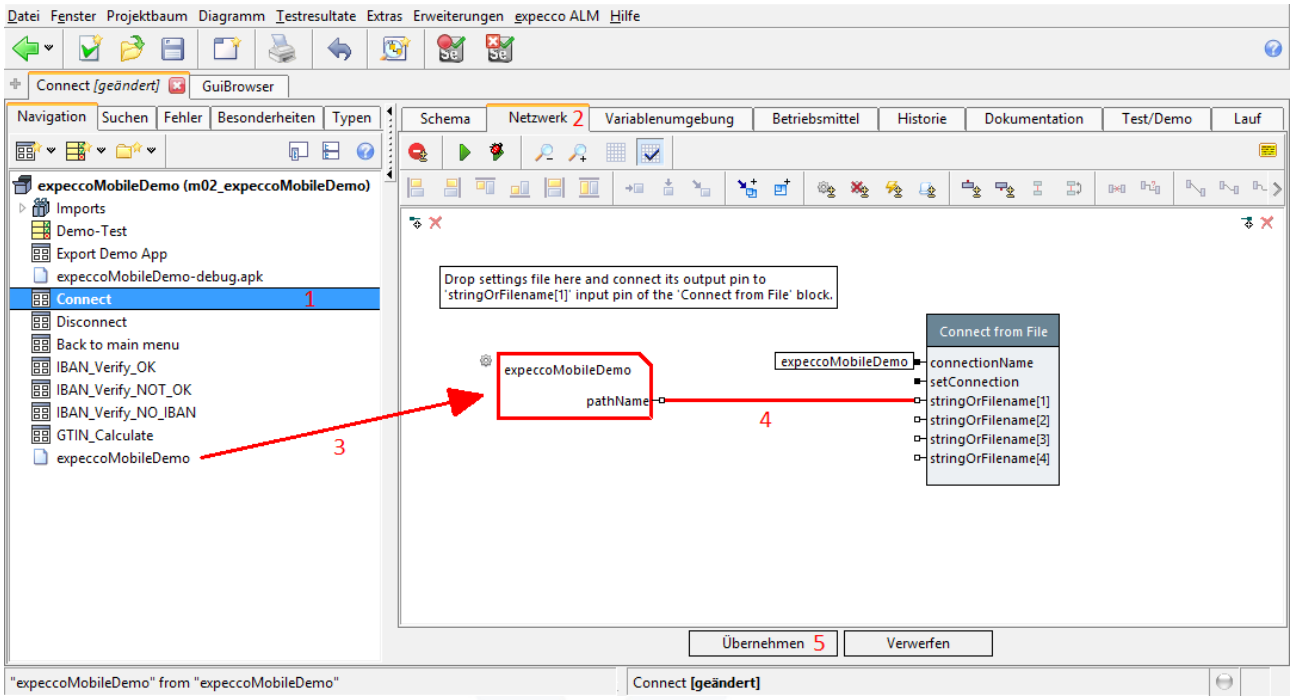

Abb. 11: Verbindungsbaustein editieren

Wechseln Sie nun zum Testplan *Demo-Test* (1) (Abb. 12). Dieser Testplan enthält bereits einige fertige Testfälle. Vor und nach der Ausführung (2) ist außerdem jeweils ein Baustein eingetragen: Der eben bearbeitete Baustein *Connect* für den Aufbau und der Baustein *Disconnect* für den Verbindungsabbau. Durch das Eintragen der beiden Bausteine an dieser Stelle geschieht der Verbindungsabbau insbesondere auch dann, wenn der Test vorzeitig abgebrochen wird, z. B. weil einer der Testfälle fehlschlägt.

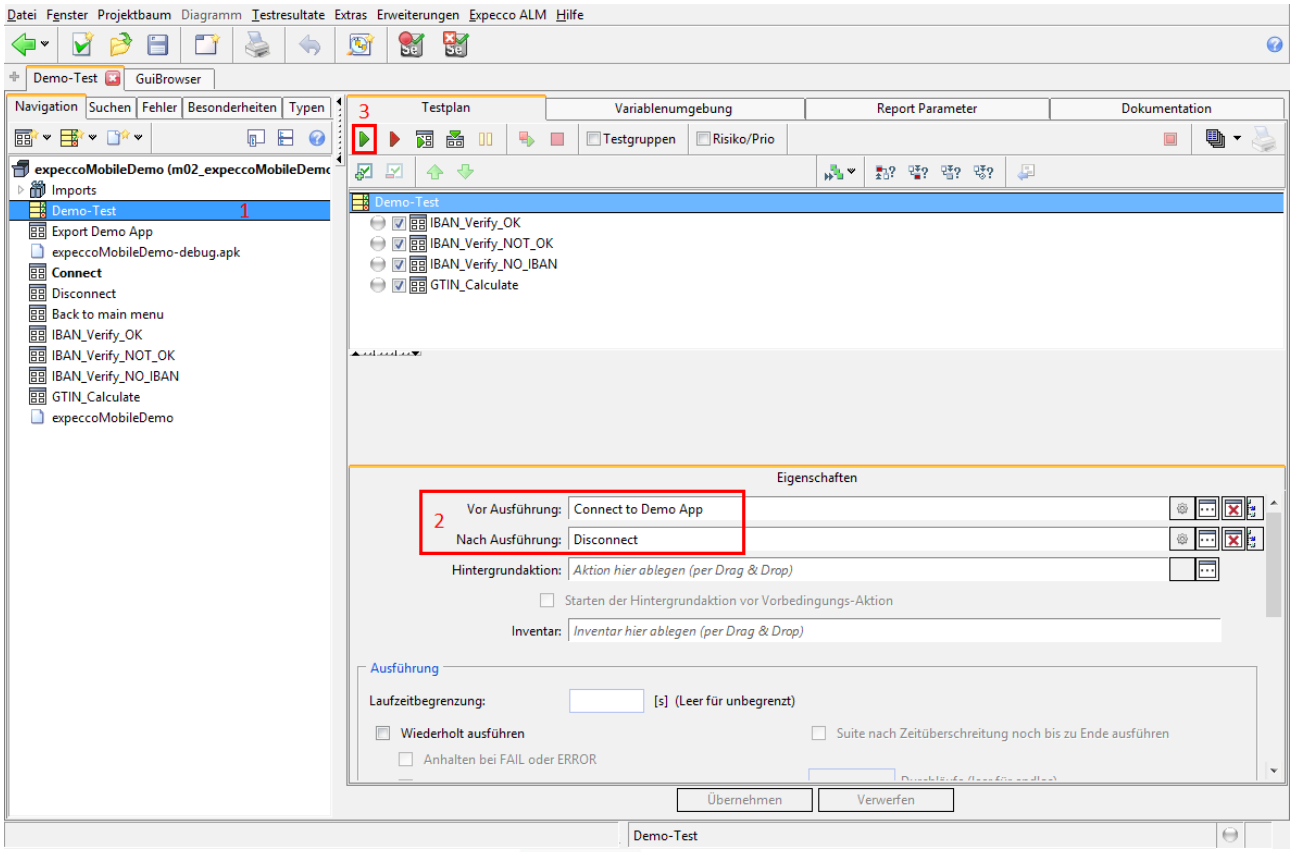

Abb. 12: Testplanausführung

Jetzt können Sie den Testplan *Demo-Test* starten, indem Sie auf den grünen Play-Knopf (3) klicken. Der Testplan sollte ohne Fehler durchlaufen.

Schritt 3: Einen Baustein mit dem Recorder erstellen

Mit Hilfe des integrierten Recorders lassen sich einfach Ausführungssequenzen aufnehmen und in einem Baustein speichern. Dafür muss eine Verbindung zu einem Testgerät bestehen, mit dessen Hilfe der Test erstellt wird.

Um eine Verbindung aufzubauen, wechseln Sie zurück zum GUI-Browser. In diesem ist nun die Verbindung eingetragen, die eben bei der Ausführung des Testplans benutzt wurde. Allerdings ist sie nicht mehr aktiv, da sie am Ende der Ausführung abgebaut wurde. Die Einstellungen sind dort aber noch eingetragen. Um die Verbindung mit dieser Konfiguration wieder aufzubauen, wählen Sie sie aus, gefolgt von einem Rechtsklick und *Verbinden*.

Warten Sie, bis die Verbindung aufgebaut ist (1) und drücken Sie dann den Aufnahme-Knopf (2), um eine Aufzeichnung zu starten (Abb. 13).

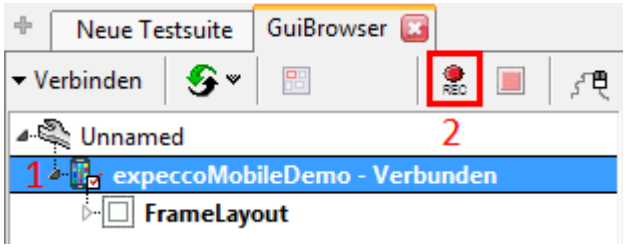

Abb. 13: Recorder starten

Es öffnet sich ein Fenster mit dem Mobile Testing Recorder (Abb. 14). Dieser zeigt einen Screenshot des verbundenen Geräts. Über diese Anzeige können Sie das Gerät fernsteuern. Dabei wird jede Aktion, die Sie ausführen, im Hintergrund aufgezeichnet.

In der oberen Menüleiste können Sie das Werkzeug auswählen (1), mit dem Sie eine Aktion eingeben möchten. Als Voreinstellung ist das Werkzeug *Auto* ausgewählt. Sie können damit bestimmte Aktionen aufzeichnen, indem Sie mit dem Mauszeiger entsprechende Gesten auf der Anzeige ausführen. Wenn Sie zum Beispiel mit der linken Maustaste lange klicken, entspricht das einem langen Antippen auf dem Touchscreen des Gerätes. Anstatt die gewünschte Aktion mit der entsprechenden Geste zu bestimmen, können Sie diese alternativ auch manuell auswählen.

Es soll nun ein neuer Test für das Erkennen korrekter GTIN-13-Codes aufgenommen werden. Klicken Sie zunächst in der Anzeige kurz auf den Button *GTIN-13 (EAN-13)* (2) der App um einen entsprechenden Klick auf dem Gerät auszulösen. Falls der Recorder danach nicht die aktuelle Ansicht der App darstellen sollte, klicken Sie im Recorder auf das Aktualisierungssymbol (3).

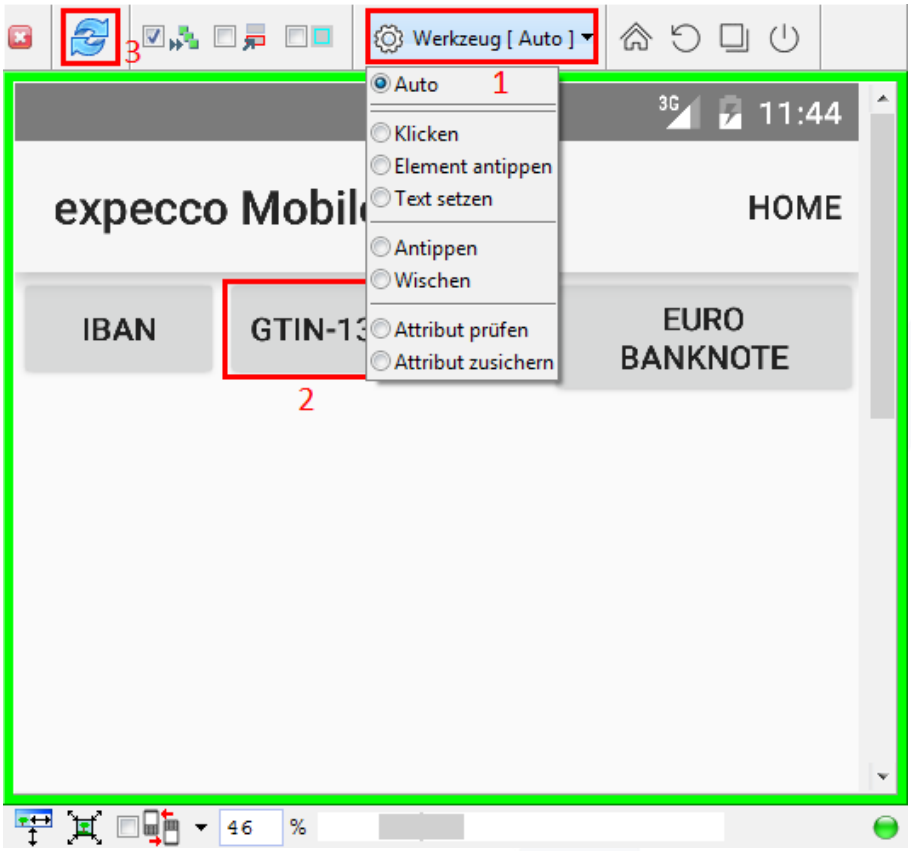

Abb. 14: Über Recorder zur GTIN-13-Activity wechseln

Anschließend soll im Eingabefeld der neuen Seite eine korrekte GTIN-13 eingegeben werden. Führen Sie dazu einen Rechtsklick auf dem Eingabefeld (1) aus und wählen Sie im Kontextmenü die Aktion *Text setzen* (2) (Abb. 15). Geben Sie in den sich daraufhin öffnenden Dialog eine beliebige gültige Artikelnummer im GTIN-13-Format ein, bspw. *4006381333986* (3). Dieser Text wird nun in der App gesetzt.

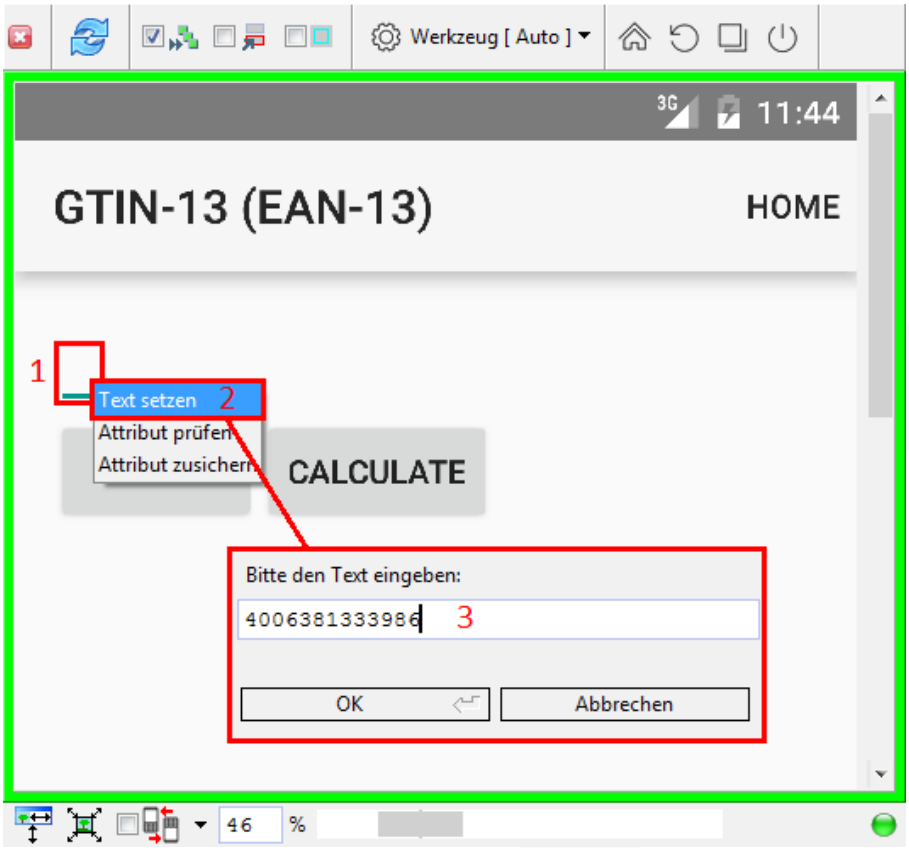

Abb. 15: GTIN-13-Code über Recorder eingeben

Klicken Sie nun auf *Verify* (1) (Abb. 16). In der App erscheint nun als Ergebnis *OK* (2). Der Test soll feststellen, ob tatsächlich dieses Ergebnis angezeigt wird. Nach einem Rechtsklick darauf können Sie im Kontextmenü die Aktion *Attribut zusichern* (3) auswählen. Wählen Sie im Dialog, der sich daraufhin öffnet, die Eigenschaft *text* (4) aus und bestätigen Sie mit *OK* (5). Dieses Mal wird keine Aktion auf dem Gerät ausgelöst, sondern nur ein Baustein aufgezeichnet, der fehlschlägt, sollte das Ergebnis vom erwarteten Wert *OK* abweichen.

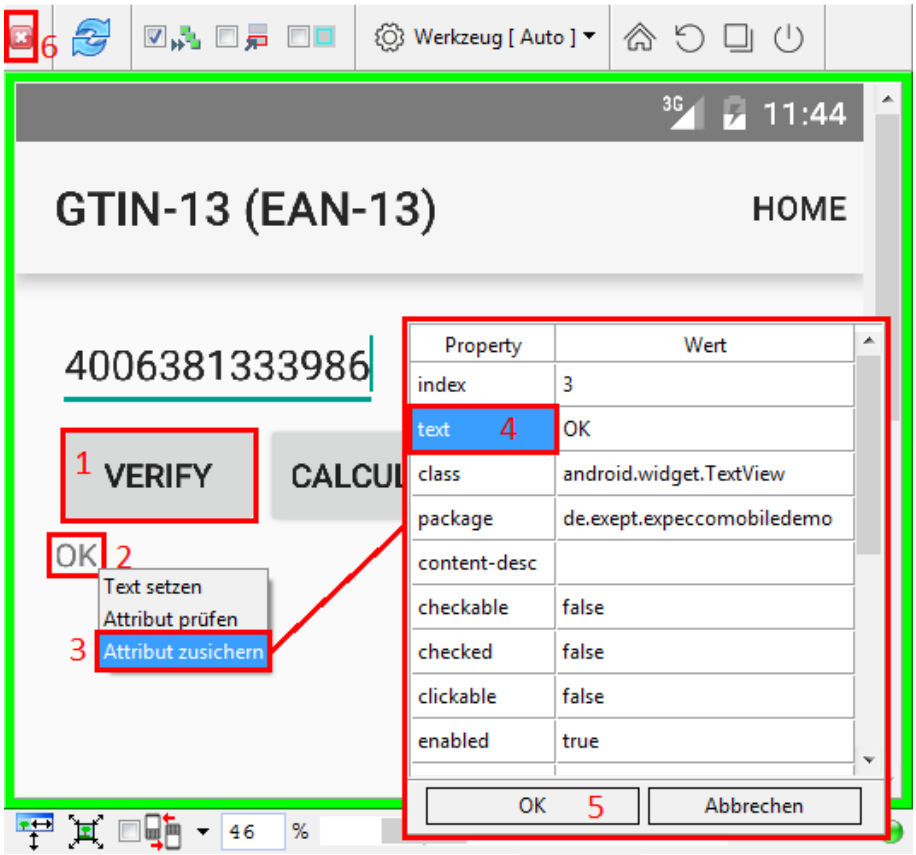

Abb. 16: Antwort der App über Recorder auslesen

Schließen Sie nun den Recorder (6). Im *Arbeitsbereich* des GUI-Browsers sehen Sie, dass für jede der aufgenommenen Aktionen ein Baustein angelegt wurde (Abb. 17). Sie können nun testen, ob sich das Aufgenommene wieder abspielen lässt. Dazu müssen Sie zunächst die App auf Ihrem Gerät in den Anfangszustand zurückbringen, indem Sie auf dem Gerät die Schaltfläche *HOME* oben rechts benutzen. Klicken Sie dann in expecco auf den grünen Play-Knopf (1). Wird alles grün, war die Ausführung erfolgreich. Erstellen Sie nun daraus einen neuen Baustein in der Testsuite, indem Sie auf das Bausteinsymbol (2) in der rechten oberen Ecke klicken. Geben Sie ihm den Namen *GTIN_Verify_OK* (3) und bestätigen Sie (4).

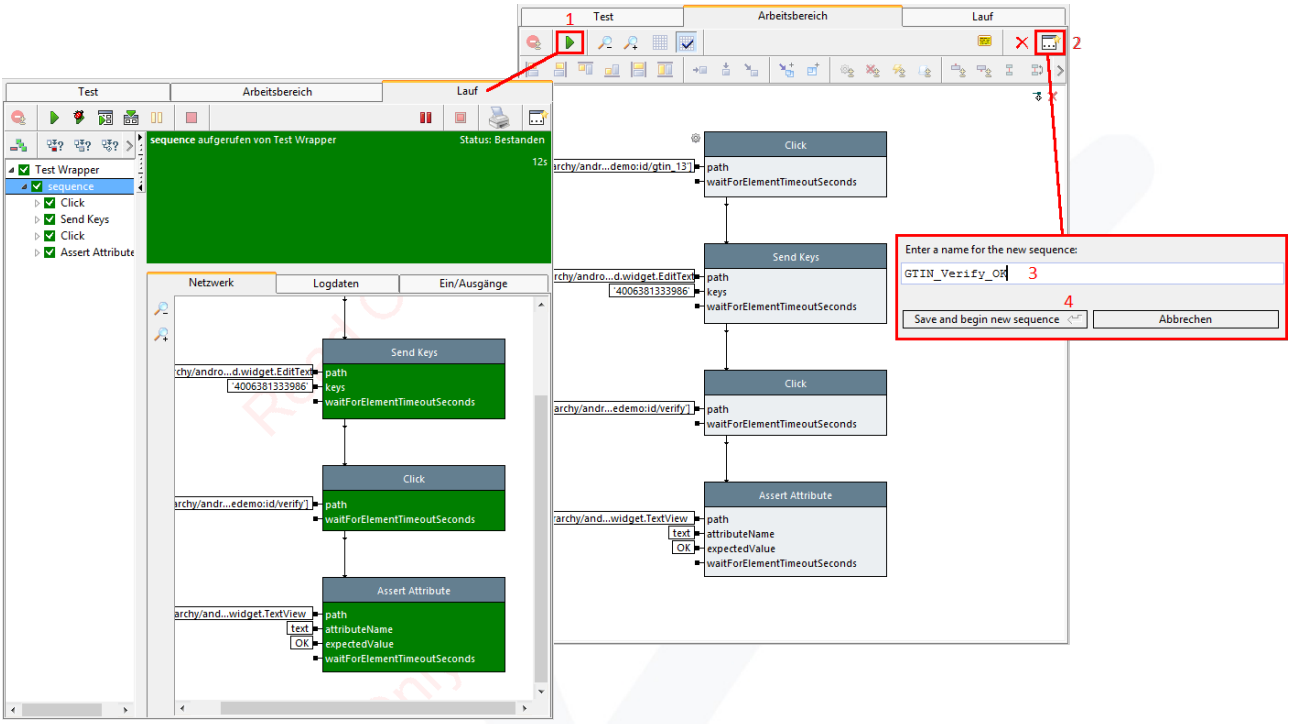

Abb. 17: Neuen Baustein aus Arbeitsbereich exportieren

Bauen Sie nun die Verbindung ab, indem Sie die Verbindung auswählen, rechtsklicken und im Kontextmenü *Verbindung abbauen* auswählen.

Wechseln Sie zurück zum Reiter der Testsuite. Dort wurde der neue Baustein angelegt. Wählen Sie wieder den Testplan *Demo-Test* aus und fügen Sie den aufgenommenen Testfall *GTIN_Verify_*OK per Drag-and-drop am Ende des Testplans hinzu. Übernehmen Sie die Änderung und starten Sie erneut. Der Testplan sollte wieder ohne Fehler durchlaufen.

Schritt 4: XPath anpassen mithilfe des GUI-Browsers

Ihr neuer Baustein sollte nun auf Ihrem Gerät fehlerfrei funktionieren, auf anderen Geräten jedoch möglicherweise nicht. Um Bausteine für eine Vielzahl von Geräten verallgemeinern zu können, muss man deren grundlegendes Funktionsprinzip verstehen: Die Ansicht der App setzt sich aus einzelnen Elementen zusammen. Dazu gehören die Schaltflächen *GTIN-13 (EAN-13)* und *Verify*, das Eingabefeld der Zahl *4006381333986* und das Ergebnisfeld, in dem *OK* erscheint, wie auch alle anderen auf der Anzeige sichtbaren Dinge. Diese sichtbaren Elemente sind in unsichtbare Strukturelemente eingebettet. Alle Elemente zusammen sind in einer zusammenhängenden Hierarchie, dem Elementbaum, organisiert.

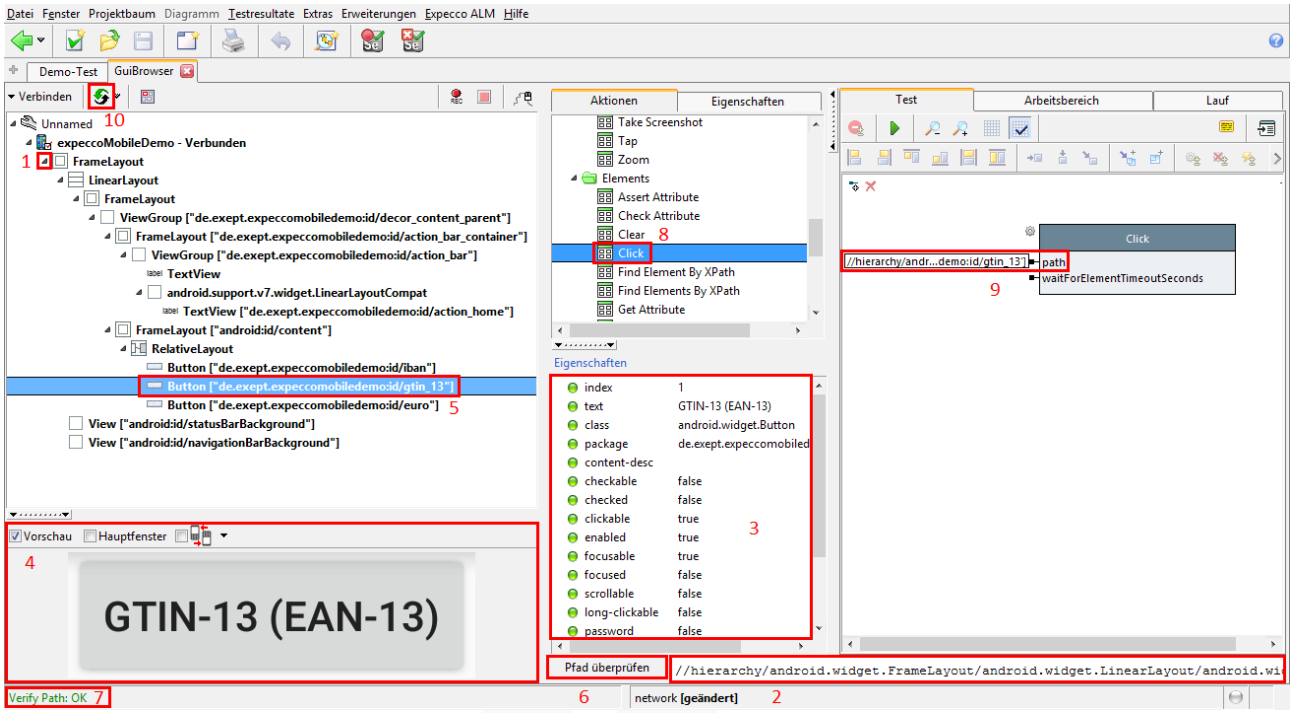

Abb. 18: Funktionen des GUI-Browsers

Sie können sich diesen Baum im GUI-Browser ansehen. Wechseln Sie dazu wieder in den GUI-Browser (Abb. 18) und starten Sie die Verbindung indem Sie den zuvor erstellten Eintrag auswählen und nach einem Rechtsklick im Kontextmenü auf den Eintrag *Verbinden* klicken. Sobald die Verbindung aufgebaut ist, können Sie den gesamten Baum aufklappen (1) (Klick bei gedrückter Strg-Taste). Er enthält alle Elemente der aktuellen Seite der App.

Ein Baustein, der nun ein bestimmtes Element verwendet, muss dieses eindeutig angeben, indem er dessen Position im Elementbaum mit einem Pfad im XPath-Format beschreibt. Dieses Format ist ein verbreiteter Web-Standard für XML-Dokumente und -Datenbanken, eignet sich aber genauso für Pfade im Elementbaum.

Wenn Sie ein Element im Baum auswählen, wird unten der von expecco automatisch generierte XPath (2) für das Element angezeigt, der auch beim Aufzeichnen verwendet wird. Oberhalb davon in der Mitte des Fensters befindet sich eine Liste der Eigenschaften (3) des ausgewählten Elements. Man nennt diese Eigenschaften auch Attribute. Sie beschreiben das Element näher wie beispielsweise seinen Typ, seinen Text oder andere Informationen zu seinem Zustand. Links unten können Sie zur besseren Orientierung im Baum die *Vorschau* (4) aktivieren, um sich den Bildausschnitt des Elements anzeigen zu lassen.

Der Elementbaum, den Sie sehen, kann sich je nach Gerät von dem hier abgebildeten unterscheiden. Es sind diese Unterschiede, die verhindern, eine Aufnahme von einem Gerät unverändert auch auf allen anderen Geräten abzuspielen: Ein XPath, der im einen Elementbaum ein bestimmtes Element identifiziert, beschreibt nicht unbedingt das gleiche Element im Elementbaum auf einem anderen Gerät. Es kann stattdessen passieren, dass der XPath auf kein Element, auf ein fal-

sches Element oder auf mehrere Elemente passt. Dann schlägt der Test fehl oder er verhält sich unerwartet.

Man könnte natürlich für jedes Gerät einen eigenen Testfall schreiben. Das brächte aber unverhältnismäßigen Aufwand bei Testerstellung und -wartung mit sich. Das Problem lässt sich auch anders lösen, da ein jeweiliges Element nicht nur durch genau einen XPath beschrieben wird. Vielmehr erlaubt der Standard mithilfe verschiedener Merkmale unterschiedliche Beschreibungen für ein und dasselbe Element zu formulieren. Das Ziel ist daher, einen Pfad zu finden, der auf allen für den Test verwendeten Geräten funktioniert und überall eindeutig zum richtigen Element führt.

Suchen Sie nun den Eintrag des GTIN-13-Buttons und wählen Sie ihn aus (5). Dessen automatisch generierter XPath (2) kann beispielsweise so aussehen:

//hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.view.ViewGroup/android.widget.FrameLayout[@resource-id='android:id/content']/android.widget.RelativeLayout/android.widget.Button[@resource-id='de.exept.expeccomobiledemo:id/gtin_13']

Er ist offensichtlich lang und unübersichtlich. Der sehr viel kürzere Pfad

//[@text='GTIN-13 (EAN-13)']*

führt zum selben Element. Sie können diesen Pfad im GUI-Browser ändern und durch *Pfad überprüfen* (6) feststellen, ob er weiterhin auf das ausgewählte Element zeigt, was expecco mit *Verify Path: OK* (7) bestätigen sollte. Der erste, sehr viel längere Pfad, beschreibt den gesamten Weg vom obersten Element des Baumes bis hin zum gesuchten Button. Der zweite Pfad hingegen wählt mit * zunächst sämtliche Elemente des Baumes und schränkt die Auswahl dann auf genau die Elemente ein, die ein *text*-Attribut mit dem Wert *GTIN-13 (EAN-13)* besitzen, in unserem Fall also genau der eine Button, den wir suchen.

Sie können solche Pfade mit Hilfe weniger Regeln selbst formulieren. Sehen Sie sich den einfachen Baum einer fiktiven Android-App in Abb. 19 an: Die Einrückungen innerhalb des Baumes geben die Hierarchie der Elemente wieder. Ein Element ist ein *Kind* eines anderen Elementes, wenn jenes andere Element das nächsthöhere Element mit einem um eins geringeren Einzug ist. Jenes Element ist das *Elternelement* des Kindes. Sind mehrere untereinander stehende Elemente gleich eingerückt, so sind sie also alle Kinder desselben Elternelements.

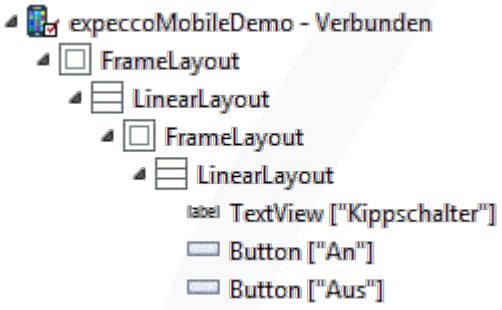

Abb. 19: Elementbaum einer fiktiven App

Ein Pfad durch alle Ebenen der Hierarchie zum TextView-Element ist nun:

//hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.TextView

Die Elemente sind mit Schrägstrichen voneinander getrennt. Es fällt auf, dass der Name des ersten Elements nicht mit dem im Baum übereinstimmt. Das oberste Element in der Hierarchie heißt immer *hierarchy*, expecco zeigt im Baum stattdessen den Namen der Verbindung an, damit man mehrere Verbindungen voneinander unterscheiden kann. Die weiteren Elemente tragen jeweils das Präfix *android.widget.*, das im Baum zur besseren Übersicht nicht angezeigt wird. Mit jedem Schrägstrich führt der Pfad über eine Eltern-Kind-Beziehung in eine tiefere Hierarchie-Ebene, d. h. *FrameLayout* ist ein Kindelement von *hierarchy*, *LinearLayout* ist ein Kind von *FrameLayout* usw. Die in eckigen Klammern geschriebenen Wörter dienen nur als Orientierungshilfe im Baum. Sie gehören nicht zum Typ.

Ein Pfad muss nicht beim Element *hierarchy* beginnen. Man kann den Pfad beginnend mit einem beliebigen Element des Baumes bilden. Man kann also verkürzt auch

//android.widget.TextView

schreiben. Der Pfad führt zum selben *TextView*-Element, da es nur ein Element dieses Typs gibt. Anders verhält es sich bei dem Pfad

//android.widget.Button.

Da es zwei Elemente vom Typ *Button* gibt, passt dieser Pfad auf zwei Elemente, nämlich den *Button*, der mit *"An"* markiert ist, und den *Button*, der mit *"Aus"* markiert ist. Es würde an dieser Stelle aber auch nicht helfen den langen Pfad von *hierarchy* aus beginnend anzugeben. Um einen mehrdeutigen Pfad weiter zu differenzieren, kann man explizit ein Element aus einer Menge wählen, indem man den numerischen Index in eckigen Klammern dahinter schreibt. Der Pfad aus dem obigen Beispiel lässt sich damit so anpassen, dass er eindeutig auf den *Button* mit der Markierung *"Aus"* weist:

//android.widget.Button[1].

Ihnen fällt sicher auf, dass der Index eine 1 ist obwohl das zweite Element gemeint ist. Das kommt daher, dass die Zählung bei 0 beginnt. Der *Button* mit der Markierung *"An"* hat also die Nummer 0 und der *Button* mit der Markierung *"Aus"* hat die Nummer 1.

Dieser Ansatz, einen expliziten Index zu verwenden, hat zwei Nachteile: Zum einen lässt sich an dem Pfad nur schwer ablesen welches Element gemeint ist, zum andern ist der Pfad sehr empfindlich schon gegenüber kleinsten Änderungen, wie zum Beispiel dem Vertauschen der beiden *Button*-Elemente oder dem Einfügen eines weiteren *Button*-Elements in der App.

Es wäre daher wünschenswert, das gemeinte Element über eine ihm charakteristische Eigenschaft wie einem Attributwert, zu adressieren. Der XPath-Standard erlaubt solche Auswahlbedingungen zu einem Element anzugeben. Angenommen, der *Button* mit der Markierung *"Aus"* hat die Eigenschaft *resource-id* mit dem Wert *off* und der *Button* mit der Markierung *"An"* hat als *resource-id* den Wert *on*, dann kann man als eindeutigen Pfad für den *"Aus"-Button*

//android.widget.Button[@resource-id='off']

formulieren. Wie an dem Beispiel zu sehen werden solche Bedingungen wie ein Index in eckigen Klammern an den Elementtyp angehängt. Der Name eines Attributes wird mit einem @ eingeleitet und der Wert mit einem = in Anführungszeichen angehängt. Ist der Attributwert global eindeutig, kann man den vorausgehenden Pfad sogar durch den globalen Platzhalter * ersetzen, der auf jedes Element passt. Das obige Beispiel mit dem GTIN-13-*Button* war ein solcher Fall.

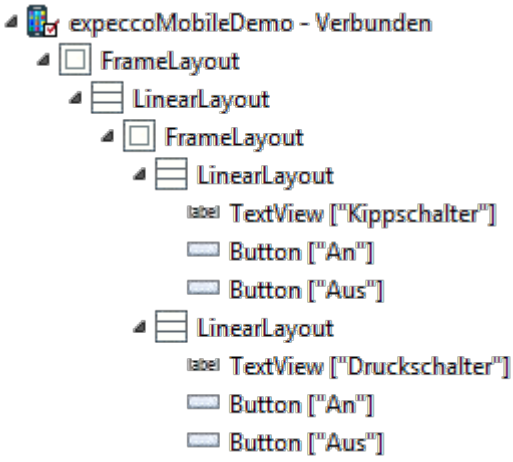

Abb. 20: Elementbaum einer fiktiven App mit Erweiterungen

Abb. 20 zeigt eine Erweiterung des Beispiels aus Abb. 19. Die App hat nun ein weiteres, nahezu identisches *LinearLayout* bekommen. Die *Buttons* sind in ihren Attributen jeweils ununterscheidbar. Deshalb funktioniert der vorige Ansatz nicht, einen eindeutigen Pfad nur mithilfe eines Attributwerts zu formulieren. Offensichtlich unterscheiden sich aber ihre benachbarten *TextViews*. Es ist möglich die jeweilige *TextView* in den Pfad mit aufzunehmen, um einen Button dennoch eindeutig zu adressieren. Ein Pfad zum *Button* mit der Markierung *"An"* unterhalb der *TextView* mit der Markierung *"Druckschalter"* kann dabei wie folgt aussehen:

//android.widget.TextView[@resource-id='push']/../android.widget.Button[@resource-id='on']

Der erste Teil beschreibt den Pfad zu der *TextView* mit der Markierung *"Druckschalter"* und der *resource-id* mit dem Wert *push*. Danach folgt ein Schrägstrich gefolgt von zwei Punkten. Die zwei Punkte sind eine spezielle Elementbezeichnung, die nicht ein Kindelement benennt, sondern zum Elternelement wechselt, in diesem Fall also das *LinearLayout*, in dem die *TextView* eingebettet ist. Im Kontext dieses *LinearLayout* ist der restliche Pfad, nämlich der *Button* mit der *resource-id* mit dem Wert *on*, eindeutig.

Der XPath-Standard bietet noch sehr viel mehr Ausdrucksmittel. Mit der hier knapp vorgestellten Auswahl ist es aber bereits möglich für die meisten praktischen Testfälle gute Pfade zu formulieren. Eine vollständige Einführung in XPath ginge über den Umfang dieser Einführung weit hinaus. Sie finden zahlreiche weiterführende Dokumentationen im Web und in Büchern.

Eine universelle Strategie zum Erstellen guter XPaths gibt es nicht, da sie von den Testanforderungen abhängt. In der Regel ist es sinnvoll, den XPath kurz und dennoch eindeutig zu halten. Häufig lassen sich Elemente über Eigenschaften identifizieren wie beispielsweise ihren Text. Will man aber gerade den Text eines Elements auslesen, kann dieser natürlich nicht im Pfad verwendet werden, da er vorher nicht bekannt ist. Ebenso wird der Text variieren, wenn die App mit verschiedenen Sprachen gestartet wird.

Jeder Baustein, der auf einem Element arbeitet, hat einen Eingangspin für den XPath. Im GUI-Browser finden Sie in der Mitte oben eine Liste von Bausteinen mit Aktionen, die Sie auf das ausgewählte Element anwenden können. Suchen Sie den Baustein *Click* (8) im Ordner *Elements* und wählen Sie ihn aus (Abb. 18). Er wird im rechten Teil unter *Test* eingefügt, der Pin für den XPath ist

mit dem automatisch generierten Pfad des Elements vorbelegt (9). Sie können den Baustein hier auch ausführen. Die Ansicht wechselt dann auf *Lauf*. Ändert sich durch die Aktion der Zustand Ihrer App, müssen Sie den Baum anschließend aktualisieren (10).

Wenn Sie in der unteren Liste eine Eigenschaft auswählen, wechselt die Anzeige der Bausteine zu *Eigenschaften*, wo Sie die eigenschaftsbezogenen Bausteine finden. Wie bei den Aktionen können Sie auch hier einen Baustein auswählen, der dann rechts in *Test* mit dem Pfad des Elements und der ausgewählten Eigenschaft eingetragen wird, sodass Sie ihn direkt ausführen können.

Versuchen Sie nun die Pfade in Ihren erstellten Bausteinen zu verallgemeinern. Sie können sich dabei an den Pfaden der bereits vorhandenen Bausteine orientieren. Starten Sie den Test erneut. Sollte der Test jetzt fehlschlagen, überprüfen Sie die Pfade noch einmal.

Falls Ihnen ein weiteres Gerät zur Verfügung steht, können Sie den Test nun auch darauf ausführen. Öffnen Sie dazu im Menü *Erweiterungen* > *Mobile Testing* > *Verbindungseinstellungen erstellen*. Sie erhalten einen Dialog ähnlich zum Verbindungsdialog. Allerdings können Sie hier nur Einstellungen erstellen und speichern aber keine Verbindung herstellen. Sie haben jedoch die Möglichkeit, einzelne Aspekte der Einstellungen zu speichern wie bspw. nur das Gerät. Wählen Sie das neue Gerät aus und klicken Sie länger auf das Symbol zum Speichern im Anhang bis sich das verzögerte Menü öffnet (Abb. 21). Wählen Sie hier *Geräte-Einstellungen speichern*. Benennen Sie den Anhang am besten nach dem Gerät. Danach können Sie den Dialog wieder schließen.

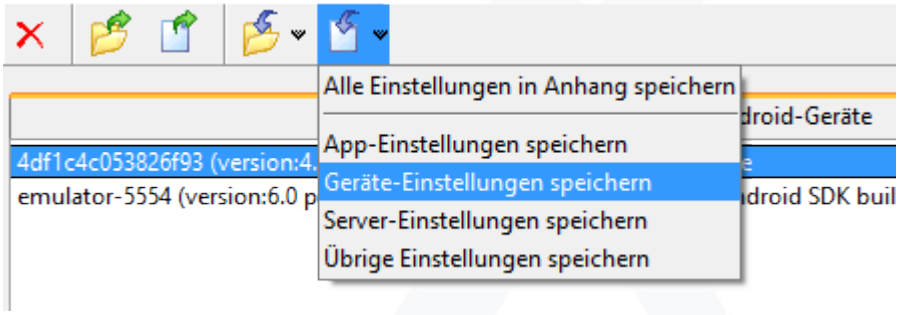

Abb. 21: Einstellungen für ein Gerät speichern

Wählen Sie den Baustein *Connect* aus und ziehen Sie die Einstellungen für das neue Gerät in dessen Netzwerk. Verbinden Sie nun dessen Ausgangspin *pathName* mit dem Eingangspin *stringOrFilename[2]* des Bausteins *Connect from File*. Der Baustein *Connect from File* liest die Angaben an den Eingangspins von oben nach unten ein, mehrfache Eigenschaften werden dabei ersetzt. In diesem Fall werden also die Einstellungen zum verwendeten Gerät ersetzt, während die übrigen Einstellungen gleich bleiben. Wenn Sie die Pfade geschickt gewählt haben, wird der Test nun auch auf dem anderen Gerät erfolgreich ablaufen.

Schritt 5: Noch einen Baustein erstellen

Falls sich gleiche Abläufe im Test wiederholen, können Sie dafür bereits erstellte Bausteine wiederverwenden oder abwandeln. Der in Schritt 3 erstellte Baustein prüft die Erkennung korrekter GTIN-13-Codes. Es fehlt noch ein Test, der umgekehrt das Erkennen eines falschen GTIN-13-Codes prüft. Die Struktur der beiden Tests ist identisch, sie unterscheiden sich lediglich in ihren Parametern. Kopieren Sie daher den Baustein *GTIN_Verify_OK* und benennen Sie die Kopie in *GTIN_Verify_NOT_OK* um. Ändern Sie die Eingabe der GTIN-13 in einen falschen Code zum

Beispiel durch Ändern der letzten Ziffer (*4006381333987)* und setzen Sie den Überprüfungswert der Ausgabe auf *NOT OK* (Abb. 22).

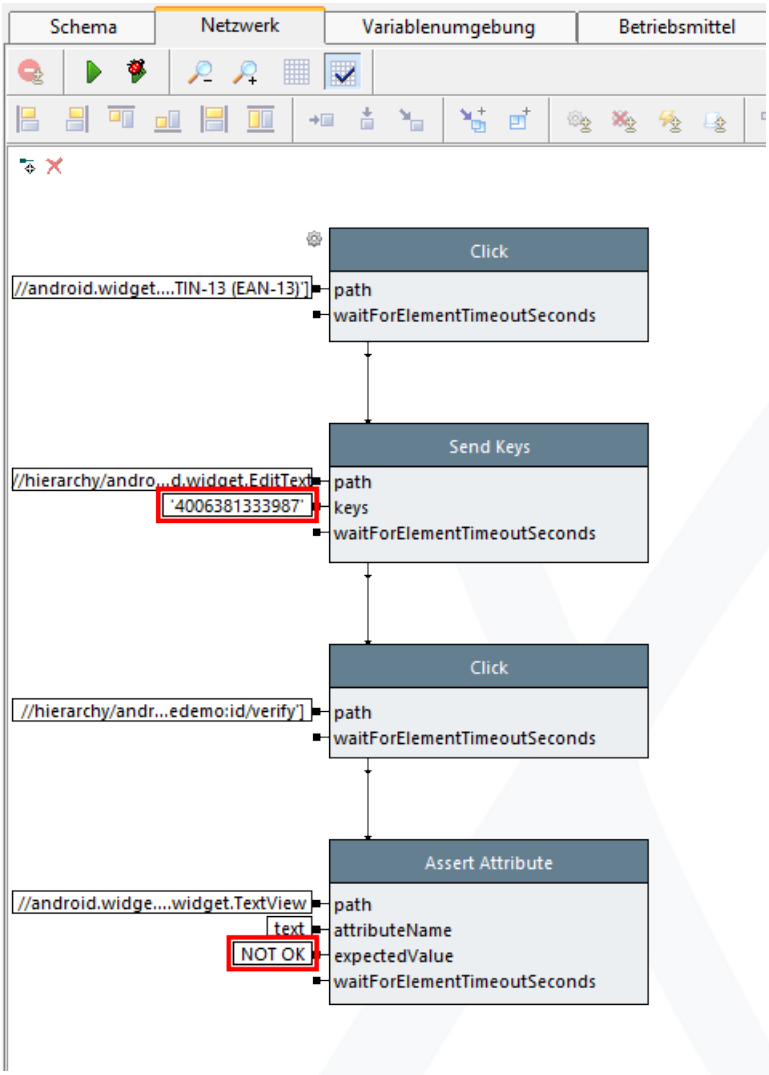

Abb. 22: Baustein editieren

Fügen Sie diesen neuen Test ebenfalls zum Testplan *Demo-Test* hinzu und setzen Sie ihn ans Ende. Führen Sie den Testplan aus, aber vergessen Sie nicht, vorher die Verbindung im GUI-Browser abzubauen.

Der neue Test wird fehlschlagen, weil Ihr aufgenommener Baustein nicht wieder zur Startseite der App zurückkehrt, die Tests aber jeweils von dort aus starten. In den anderen Bausteinen ist dies bereits berücksichtigt; sie führen abschließend immer den Baustein *Back to main menu* aus. Sie sehen das, indem Sie einen der anderen Bausteine, z. B. *GTIN_Calculate,* auswählen und auf seine Schema-Ansicht wechseln. Dort steht der Baustein *Back to main menu* im Feld *Nach Ausführung* (Abb. 23). Wie beim entsprechenden Feld beim Testplan wird auch dieser Baustein immer am Ende ausgeführt, unabhängig davon, ob der Test erfolgreich verläuft oder abbricht. Ergänzen Sie diesen Eintrag nun in Ihren Bausteinen *GTIN_Verify_OK* und

GTIN_Verify_NOT_OK. Wählen Sie dazu den Baustein aus und ziehen Sie in der Schema-Ansicht den Baustein *Back to main menu* einfach auf das Eingabefeld *Nach Ausführung*. Nun können Sie den Testplan starten und alle Tests sollten wieder problemlos ausgeführt werden.

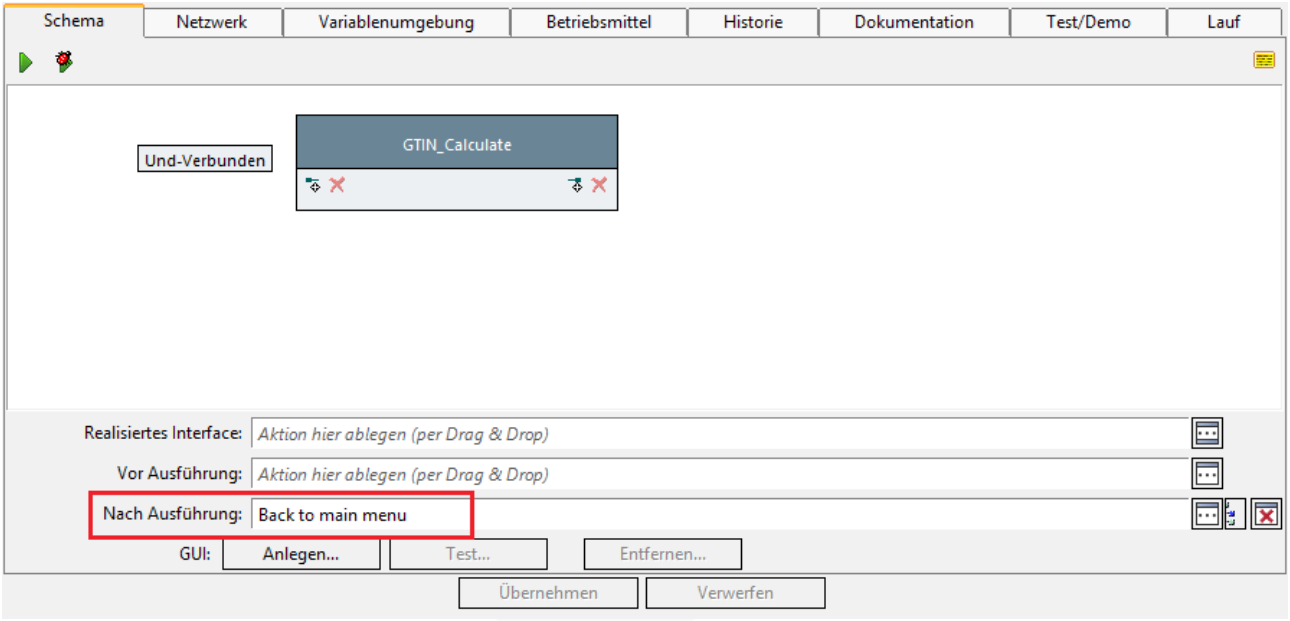

Abb. 23: Nach-Ausführungs-Baustein setzen

Schritt 6: Test vervollständigen

Für die Activity IBAN sind bereits alle Antwortmöglichkeiten der App mit Testfällen abgedeckt. In der GTIN-13-Activity werden ein korrekter und ein fehlerhafter Code getestet und eine Prüfziffer berechnet, das Verhalten der App bei Eingaben falscher Länge wird aber bisher nicht getestet (Bei Verify *Input must be exactly 13 digits.* und *…12 digits.* bei Calculate). Die Activity zum Prüfen der Seriennummern von Eurobanknoten wird noch gar nicht getestet. Wie bei der IBAN können hier drei Fälle auftreten: eine korrekte Seriennummer wurde eingegeben (Antwort: *OK*), eine falsche Seriennummer wurde eingegeben (Antwort: *NOT OK*) oder die Angabe entspricht nicht dem Format (Antwort: *A serial number consists of 12 characters with the first one or two being capital letters (A-Z).*). Sie können die Testabdeckung jetzt noch erweitern, indem Sie entsprechende Testfälle erstellen. Die Bausteine dafür können Sie wie in Schritt 3 mit dem Recorder erstellen und wie in Schritt 4 die XPaths verallgemeinern. Wenn Sie mit dem grundsätzlichen Umgang mit expecco vertraut sind, können Sie Bausteine natürlich auch ohne Recorder erstellen, indem Sie sie manuell aus vorhandenen Bausteinen der Bibliothek zusammensetzen. Nach Bedarf können Sie auch beide Vorgehensweisen kombinieren.

Beachten Sie, dass die hier vorgestellten Testfälle jeweils nur einzelne Eingaben prüfen. Wenn Sie Testfälle für Ihre eigenen Apps schreiben, wollen Sie vermutlich engmaschiger testen, indem Sie noch weitere unterschiedliche Werte eingeben, die insbesondere auch Randfälle enthalten.

Weiterführende Links

Wiki zu expecco, hier finden Sie ausführliche Informationen zum Erstellen von Tests mit expecco:
http://doc.expecco.de/wiki2.x/

Wikiartikel zum Mobile Testing Plugin mit ausführlicheren Erklärungen zur Benutzung:
http://doc.expecco.de/wiki2.x/Mobile_Testing_Plugin

Wikiartikel über die Benutzung des GUI-Browsers:
http://doc.expecco.de/wiki2.x/Expecco_GUI_Tests_Extension_Reference

Informationen zu Appium:
http://appium.io