Auch hier wird jeweils ein Array pro *Constraint* angelegt. Das zweite View wird jeweils mit 50 Pixeln für Höhe und Breite definiert. Anschließend werden auch diese beiden *Constraints* mit der *addConstraints*-Methode dem zweiten View hinzugefügt. Zuletzt geht es an die Positionierung der beiden Views. Hierfür müssen noch einmal zwei *Constraints* angelegt werden, die diesmal allerdings für beide Views gelten:

```
let views_constraint_HPos:NSArray =
NSLayoutConstraint.constraints(withVisualFormat: "H:|-30-[viewBlue]-40-
[viewGreen]", options: NSLayoutFormatOptions(rawValue: 0), metrics: nil,
views: viewsDictionary) as NSArray
let views_constraint_VPos:NSArray =
NSLayoutConstraint.constraints(withVisualFormat: "V:|-20-[viewBlue]-10-
[viewGreen]", options: NSLayoutFormatOptions(rawValue: 0), metrics: nil,
views: viewsDictionary) as NSArray

view.addConstraints(views_constraint_HPos as! [NSLayoutConstraint]
)

view.addConstraints(views_constraint_VPos as! [NSLayoutConstraint]
)
```

Mit dem ersten Parameter werden die zusätzlichen Constraints übergeben. Sie enthalten nun Informationen sowohl für das blaue als auch für das grüne View. Mit *H* wird die horizontale Ausrichtung beschrieben. Das blaue View wird 30 Pixel vom Rand und das grüne View 40 Pixel ausgehend vom blauen View eingefügt. Für die vertikale Ausrichtung werden 20 Pixel vom Rand für das blaue View und 10 Pixel (vom blauen View aus) festgelegt. Startet man die App jetzt erneut, so werden die Views an den vordefinierten Punkten gezeichnet, egal in welcher Lage sich das Gerät befindet. Eine detaillierte Einführung in die VFL finden Sie in der Apple-Dokumentation, siehe:

https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/AutolayoutPG/VisualFormatLanguage/VisualFormatLanguage.html

5.6 Workshop – Passwortverwaltung – Teil 1

Bisher haben Sie mit Ausnahme des Beispiels im zweiten Kapitel nur Apps geschrieben, die dazu dienten, eine bestimmte Funktion auszuprobieren. Mit dem nun folgenden Workshop wird sich das ändern. In ihm soll in mehreren Teilen eine »echte« App programmiert werden. Da das ganze Projekt etwas umfangreicher ist und mehrere Themengebiete anschneidet, die bis zu diesem Zeitpunkt noch nicht behandelt worden sind, ist es in insgesamt sechs Teile aufgespalten.

Im ersten Teil befassen wir uns mit der Planung der App und mit der Umsetzung der ersten benötigten Klassen. Die erste Version der App kann noch nicht viel. Sie lässt sich zwar starten, aber eine sinnvolle Funktion erfüllt sie nicht.

5.6.1 Planung der App

Das Ziel des Workshops ist die Programmierung einer Passwortverwaltung. Die wesentlichen Funktionen der App sind:

- Anzeige einer Übersicht (Liste) mit den Namen bzw. einem Hinweis zum Passwort
- eine Detailansicht, in der das Passwort sowie dessen Name und eine Bemerkung angezeigt werden
- ein Dialog, in dem das Passwort angelegt werden kann. Beim Anlegen soll es die Möglichkeit geben, Einfluss auf die Länge des Passworts sowie auf die verwendeten Zeichnen zu nehmen. Außerdem soll das Passwort sowohl automatisch erstellbar als auch manuell änderbar sein.

Das sind grob die Anforderungen an die App. Jetzt könnte man einfach drauflosprogrammieren. Der Umfang der App würde das sicherlich noch zulassen. Aber es ist sicherlich nicht schlecht, sich bereits vor der ersten Programmierung ein wenig mehr Gedanken zur App zu machen: Wie soll sie aussehen? In welcher Reihenfolge sollen die Views aufgerufen werden?

Es gibt so einige Fragen, die sich bereits im Vorfeld klären lassen. Nun könnten Sie hergehen und die Entwürfe auf ein Blatt Papier kritzeln. Das ist aber vielleicht doch etwas umständlich und wenig professionell. Vielleicht erkennen Sie in der Designphase, dass ein Control oder ein Dialog an einer anderen Stelle besser untergebracht ist. Aus diesem Grund sollten Sie auch bei der Planung einer App auf Programmunterstützung zurückgreifen.

Mit sogenannten *Mockup*-Programmen kann man die Oberflächen (nicht nur von iOS-Apps) bequem planen und erhält so schon vor der Programmierung der App einen visuellen Eindruck von dem Programm. Es gibt unterschiedliche Mockup-Programme. Einige sind kostenpflichtig, andere stehen unter der GNU General Public License.

Evolus Pencil ist so ein Programm. Es steht unter der GNU-Lizenz und kann unter folgendem Link: *https://code.google.com/p/evoluspencil/* heruntergeladen werden. Das Programm ist für unterschiedliche Plattformen (Windows, Linux und OS X) verfügbar. Mit *Pencil* können neben Mockups auch Diagramme erstellt werden. Die verfügbaren Grafiken zum Mockup von iOS-Apps entsprechen zwar nicht der neuesten GUI-Version (iOS 7 und höher), sie genügen aber, um eine App zu planen. Ein neues Projekt wird über das *Document*-Menü und dann durch Auswahl des Menüpunkts *New Document* angelegt. Die Bedienung des Programms ist weitestgehend selbsterklärend. Die Grafiken zum Mockup für iOS sind unter den Begriffen *iOS UI Stencils* und *iOS Wireframe* zu finden. Sie entsprechen weitestgehend den unter iOS verfügbaren Controls. Das bedeutet: Für (fast) jedes Control unter iOS finden Sie hier ein entsprechendes Gegenstück.

Die Erstellung eines Mockups geht relativ einfach. Nachdem Sie sich für einen Formfaktor (iPhone oder iPad) entschieden haben, positionieren Sie die entsprechende Grafik des gewählten Geräts auf dem Dokument und fügen dann via Drag & Drop alle weiteren gewünschten Elemente hinzu.

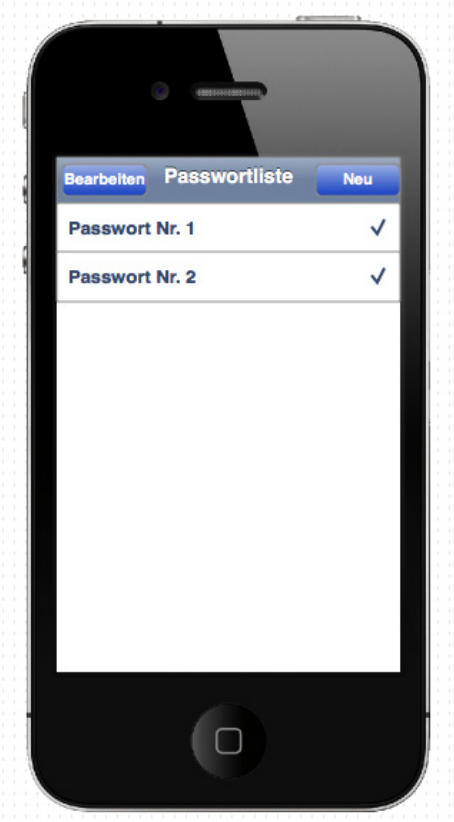

Abb. 5–47 *Planung der App – Übersicht*

Die App besteht im Wesentlichen aus drei Views, die es ermöglichen, ein Passwort auszuwählen, einzusehen und anzulegen. In Abbildung 5–47 ist ein Entwurf der Passwortliste zu sehen. Dieses View soll direkt nach dem Start der App zu sehen sein und dient somit als Einstiegspunkt in die App. Von diesem View aus lassen sich alle anderen Funktionen der App erreichen. Um ein gespeichertes Passwort einzusehen, wird der Benutzer den Eintrag zum Passwort in der Übersicht antippen. Anschließend geht es in die Detailansicht, in der das Passwort sowie zusätzliche Informationen zu ihm einsehbar sind.

Abb. 5–48 *Planung der App – Detailansicht*

In diesem View hat der Anwender ansonsten keine Möglichkeiten zur Interaktion. Möchte er zurück zur Übersicht, so muss er die entsprechende Schaltfläche betätigen (siehe Abb. 5–48).

Die letzte Station innerhalb der App ist der View zum Anlegen eines neuen Passworts. Dieses View wird aufgerufen, nachdem innerhalb der Übersicht die Schaltfläche mit der Bezeichnung *Neu* betätigt wurde. In diesem View hat der Anwender die Möglichkeit, ein neues Passwort zu generieren und auch auf dessen Generierung Einfluss zu nehmen.

Zur Einflussnahme kann er die Länge des Passworts über ein Slider-Control festlegen. Außerdem kann der Anwender über das Switch-Control bestimmen, ob das Passwort nur Großbuchstaben enthalten soll oder nicht. Im folgenden TextField wird das Passwort nach seiner Erstellung angezeigt und kann ggf. noch manuell nachbearbeitet werden. Erzeugt wird das Passwort natürlich, nachdem die entsprechende Schaltfläche betätigt wurde. Im TextField-Control unterhalb des Labels *Name* kann eine Bezeichnung für das Passwort erfasst werden, unter der es in der Übersicht angezeigt wird. Außerdem kann im TextView-Control *Bemerkung* noch eine etwas ausführlichere Beschreibung zum Passwort eingege-

Abb. 5–49 *Planung der App – Neuanlage*

ben werden. Ganz zuletzt hat der Anwender die Möglichkeit, das Passwort nach Generierung und Bearbeitung des Datensatzes zu speichern. Hierzu wird der Button mit der gleichlautenden Bezeichnung verwendet. Die Funktionalität der App ist damit komplett abgesteckt. Jetzt kann es an den ersten Teil der Umsetzung gehen.

5.6.2 Umsetzung des Projekts – Teil 1

Im ersten Teil des Projekts werden zwei Punkte umgesetzt. Es werden alle erforderlichen Views im Storyboard angelegt, und es wird die grundsätzliche Navigation in diesem Teil implementiert. Außerdem wird die Klasse zur Generierung des Passworts programmiert.

Jedes Projekt beginnt mit der Auswahl einer passenden Projektvorlage. Hierbei sollte man berücksichtigen, dass die gewählte Projektvorlage bereits möglichst viele der später benötigten Funktionen automatisch erzeugt. Die Passwortverwaltung zeigt dem Anwender die Passwörter im ersten View in Form einer Liste an, um von diesem Punkt aus in andere View zu verzweigen. Es sollte also

eine Vorlage gewählt werden, die diese Form der Anzeige begünstigt. Von den in Kapitel 1 vorgestellten Vorlagen kommt nur eine als Auswahl in Betracht. Die *Master-Detail-Application-Vorlage* entspricht bereits zum Teil den Vorgaben und dient aus diesem Grund als Ausgangsbasis für das Projekt.

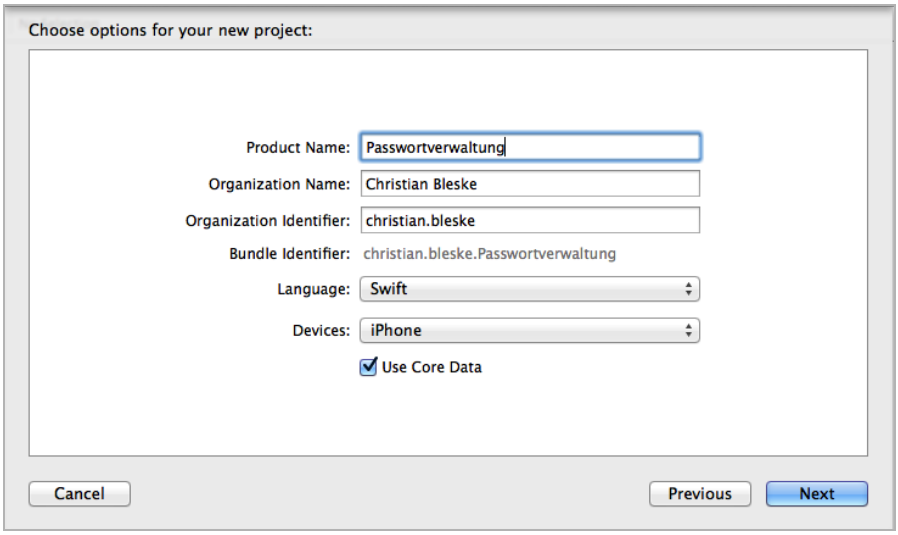

Abb. 5–50 *Neuanlegen des Projekts »Passwortverwaltung«*

Die Einstellungen des Projekts bedürfen sicherlich, mit einer Ausnahme, keiner weiteren Erläuterung. Der gewählte Produktname sowie der Name der Organisation und der Identifier sind klar, auch bei der Auswahl der Sprache gibt es sicherlich keine Fragen. Das Projekt wird erst einmal nur für das iPhone konfiguriert, das kann aber später auch problemlos geändert werden.

Einzig die aktivierte Option *Use Core Data* ist zu diesem Zeitpunkt unklar. *Core Data* ist ein Framework und stellt Klassen bereit, die die Speicherung und auch das Laden von Objekten ermöglichen. Dieses Framework wird im Detail in Kapitel 8 besprochen. Da aber die Passwortverwaltung dieses Framework bzw. Klassen daraus verwendet und die Option *Use Core Data* die einfache Verwendung ermöglicht, muss dieser Punkt beim Anlegen des Projekts aktiviert werden. Möglich wäre auch ein späterer Einbau von Core Data in das Projekt, aber so ist es bequemer. Nach dem Anlegen des Projekts müssen einige Punkte abgearbeitet werden:

1. Anpassung des (generierten) Codes in den Klassen *MasterViewController* und *DetailViewController*
2. Erweiterung des Storyboards um ein *View* und ein *Segue* sowie Anpassung des *Master-Views*
3. Hinzufügen einer neuen Swift-Datei und Implementierung der Klasse *GeneratorViewController*