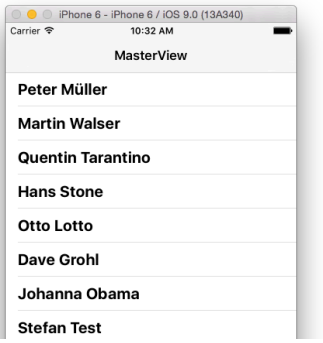


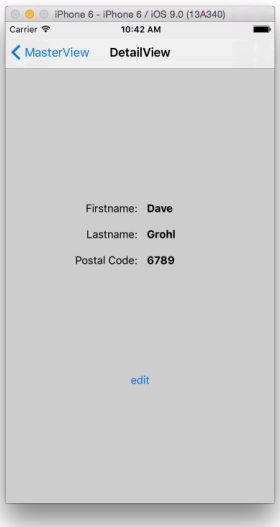
Übung 4: Tabellen, Navigation & Textfelder

Diese Aufgabe hat drei inhaltliche Schwerpunkte. Einerseits werden Tabellen im iOS angeschaut, dies umfasst insbesondere die Klasse `UITableViewController`, sowie die beiden Protokolle `UITableViewDelegate` und `UITableViewDataSource`. Zweitens geht es hierarchische Navigation und die Klasse `UINavigationController`. Drittens werden editierbare Textfelder verwendet und die dazu notwendigen iOS-Mechanismen angeschaut.

1. **Neues Projekt „EditList“**
   Erstellen sie in Xcode ein neues iOS-Projekt, wählen sie „Master-Detail App“, Produktname „EditList“, Gerät iPhone, keine Verwendung von Core Data. Starten sie das Programm, sie sehen eine Tabelle und wenn sie auf das „+“ rechts oben klicken, werden neue Zellen eingefügt. Wenn sie auf eine Zelle klicken, erscheint eine Detail-View. Studieren sie den Aufbau ihrer App im Storyboard: hier werden in einen SplitViewController zwei NavigationController verlinkt, da die App auch auf einem iPad gut aussehen soll. Bei einem NavigationController wird der MasterViewController (eine Subklasse von `UITableViewController`) als `RootViewController` gesetzt. Und dem andern NavigationController wird die DetailView als `RootViewController` gesetzt.

2. **MasterView: Tabelle mit Personen-Daten vom DataProvider**
   Nun wollen wir in dieser Tabelle die Personen aus der ersten Übung auflisten. Importieren sie (Kontextmenu: „Add Files to ...“) dazu in dieses neue Projekt die beiden Klassen `PersonData` und `DataProvider` aus der ersten Übung (Stichwort: Code-Reuse :-). Wählen sie, dass die Dateien ins Projektverzeichnis kopiert werden sollen. Implementieren sie dann in der MasterViewController-Klasse die dazu notwendigen Methoden aus dem Protokoll `UITableViewDataSource` (welches die von der `MasterViewController` als Subklasse von `UITableVieweController` schon adoptiert wird) so, dass die Tabelle „weiss“, wie viele Zeilen (= Anzahl Personen) es gibt, und welcher Name auf welcher Tabellen-Zeile steht. Das in dieser Klasse vorgenerierte Property `objects` können sie komplett entfernen - wir haben ja unseren DataProvider als Daten-Lieferent! Ebenfalls löschen sollen sie die Methoden `insertNewObject` und die beiden Methoden `tableView:canEditRowAtIndexPath:` und `tableView:commitEditingStyle: forRowAtIndexPath:` aus den Protokoll `UITableViewDataSource`. Entfernen sie im weiteren den `addButton` und das `editButtonItem` aus der Klasse `MasterViewController`. Setzen die dem MasterViewController als Titel (Property `title`) „MasterView“. Die Tabelle sollte dann in etwa so aussehen, wie im Screenshot oben rechts ersichtlich.

3. **DetailView mit Personen-Daten**
   Auf der DetailView sollen neu nicht mehr statische Daten, sondern die Daten der angewählten Person erscheinen. Löschen sie beim DetailViewController die beiden Methoden `detailItem:` und `configureView`. Modifizieren sie im Storyboard die DetailView, so dass dort Vor- und Nachnamen, sowie die PLZ



angezeigt werden, wie in dem Screenshot rechts ersichtlich, inkl. der dazu notwenigen Outlets. Was jetzt noch fehlt, ist die Übergabe der anzuzeigenden Daten von der MasterView an die DetailView. Im Code ist eine Möglichkeit dazu vorgezeichnet, der DetailViewController hat nämlich ein Property `detailItem`, ändern sie diese ab zu einem Property `person` vom Typ `PersonData`. Passen sie den Code in der Klasse `DetailViewController` entsprechend an. Modifizieren sie dann insbesondere die Methode `prepareForSegue:sender:` in der MasterViewController-Klasse dahingehend, dass dem DetailViewController via dessen neuen Property `Person` die gewünschte Person übergeben wird. Stellen sie sicher, dass die Daten der gewählten Person auf der DetailView auch angezeigt werden, indem sie die aktuellen Personen-Daten in der Methode `viewWillAppear:` vom DetailViewController auch anzeigen.

4. **Modale EditView auf der DetailView**
   Zum Abschluss sollen die Personen-Daten editiert werden können. Neu soll es auf der DetailVew einen Edit-Knopf geben. Wird dieser gedrückt, erscheint als **modale View** (also insbesondere ohne Back-Button links oben!) eine „EditView“, siehe Bilder rechts. Fügen sie dazu also ihrem Projekt im Storyboard neu eine EditView hinzu und verbinden sie diese mit einem modalen Segue. Die EditView soll ungefähr aussehen wie auf den Screenshots rechts, verwenden sie entsprechend `UILabels` und `UITextFields` und stellen sie die notwendigen `IBOutlets` zur ViewController-Klasse her. Zur Übergabe der Daten von der DetailView an die EditView soll derselben Mechanismus verwendet werden wie zwischen der MasterView und der DetailView, d.h. das soll in der Methode `prepareForSegue:sender:` passieren. Die neu zu erstellende `EditViewController`-Klasse hat also ein Property `person`, um sich die aktuelle Person zu merken.
   Hinweis: Beim Klick auf ein `UITextField` erscheint automatisch eine Tastatur. Diese verschwindet, indem auf dem entsprechendem TextField-Objekt die Methode `resignFirstResponder` aufgerufen wird. Die Tastatur soll verschwinden, wenn auf der Tastatur die Enter-Taste gedrückt wird. Hinweis: Wenn auf der Tastatur die Enter-Taste gedrückt wird, dann wird dem `delegate` des entsprechenden TextFields die Nachricht `textFieldShouldReturn:` geschickt. Wenn der save-Knopf gedrückt wird, sollen die geänderten Daten gespeichert und soll zur DetailView zurückgekehrt werden (mittels `dismissViewControllerAnimated:completion:`).
   Implementieren sie eine entsprechende IBAction-Methode `saveAndExitButtonPressed`. Damit haben sie eine einigermassen umfangreiche iPhone-App programmiert, welche sich über drei Bildschirme verteilt! ☺