Aufgabe 7: MegaCRM

Ziel

In dieser Aufgabe entwickeln wir nun den iOS Client für MegaCRM.

Aufgabe

1. Kopieren Sie die Vorlage für den MegaCRM Client

2. Schauen Sie sich kurz die Klasse `CustomerService` an. Diese implementiert die REST Server-Schnittstelle und arbeitet mit einem Delegate Pattern um mit uns zu kommunizieren (`CustomerServiceDelegate`).

3. Erstellen Sie nun wie in Aufgabe 6 gelernt, eine Tabellenansicht mit einer Detailansicht der Kunden im *Storyboard*. Vergessen Sie nicht die Tabellenansicht in einen *Navigation Controller* einzubetten damit die *Segues* funktionieren.

4. Erstellen Sie zu den beiden *Views* auch entsprechende *View Controller*. Vergessen Sie nicht diese auch mit der *View* zu verbinden im *Identity Inspector*.

5. Verwenden Sie nun im *View Controller* der Übersicht den `CustomerService` um an eine Liste mit Kunden zu kommen und diese anzuzeigen. Dafür verwenden wir die in Aufgabe 6 kennen gelernten *Data Source Delegate* Methoden.

6. Designen Sie nun die Zelle der Tabelle. Sie haben hier viele Möglichkeiten... achten Sie aber darauf, dass eine Tabellenzelle immer kompakt sein sollte.
   Tipp: Um an Elemente einer *Custom-Styled* Zelle zu gelangen müssen sie über *Tags* arbeiten. Im *Attributes Inspector* können sie jedem Element ein Tag geben und im *Controller* können Sie auf der Zelle mit `viewWithTag:` an dieses Element gelangen.

7. Implementieren Sie den *Segue* zur Detailansicht. Sie müssen hier den selektierten Kunden an die Detailansicht weitergeben, damit dieser dort angezeigt werden kann. Dies geschieht in der Methode – `(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender.`

8. Erstellen und designen Sie die Detailansicht mit den relevanten Informationen. Suchen Sie sich dafür eine Handvoll Kundenattribute aus. Die Elemente in der *View* müssen über `IBOutlets` im *Controller* zur Verfügung stehen damit sie dort gefüllt werden können.

Kunde bearbeiten

9. Als nächstes möchten wir einen Kunden bearbeiten können. Dafür brauchen wir rechts oben einen *Edit* Button in der Detailansicht. Dieser wird vom System zur Verfügung gestellt und kann wie folgt angezeigt werden:

```
self.navigationItem.rightBarButtonItem =
                                        self.editButtonItem;
```

   Dieser Button wechselt zwischen *Edit* und *Done* und ruft jedes Mal die Methode - `(void)setEditing:(BOOL)editing animated:(BOOL)animated` auf welche Sie im *ViewController* überschreiben können.

10. Um über Änderungen in den Textfeldern informiert zu werden müssen wir den *Controller* als *Delegate* bei den Textfeldern angeben. Dies geschieht genau gleich wie das Verbinden der *Outlets* (*Ctrl* Drag&Drop vom Feld auf das *ViewController* Symbol) nur, dass wir dieses Mal *delegate* auswählen.

11. Der *Controller* der Detailansicht muss nun noch das Protokoll UITextFieldDelegate implementieren und in den entsprechenden Delegate Methoden die Änderungen zurück in den Kunden schreiben.

12. Speichern Sie die geänderten Werte mit dem Service. Vergessen Sie dabei nicht, dass auch unsere Übersichtstabelle aktualisiert werden muss.

Kunde erstellen

13. Erstellen Sie einen Button in der Navigationsleiste der Übersichtsview einen Button (*Bar Button Item*)

14. Um einen neuen Kunden zu erstellen brauchen wir nun einen zweiten *Segue* von der Übersicht- auf die Detailansicht. Ziehen Sie dafür den *Segue* vom neu erstellten Button auf unsere Detailansicht.

15. Die beiden *Segues* brauchen unterschiedliche *Identifier* um sie in der Methode `–prepareForSegue:` unterscheiden zu können.

16. Für den zweiten *Segue* müssen Sie nun ein neues Kunden Objekt erstellen und der Detailansicht mitteilen, dass wir uns in einer anderen Situation befinden, damit diese beim Speichern nicht nicht ein bestehenden Kunden aktualisieren will sondern einen neuen anlegt.

Mögliche Erweiterungen und Verbesserungen

Hier finden Sie eine nicht abschliessende Liste mit Ideen wie die App verbessert oder erweitert werden könnte:

- Alphabetisch sortierte Tabelle
- Ist das Arbeiten mit der Tabelle effizient? Wird immer die ganze Tabelle bei Änderungen neu geladen oder nur jene Zellen die davon betroffen sind?
- Animationen beim Einfügen, Ändern und Löschen von Zellen/Kunden.
- Abschnitte (*Sections*) in der Tabelle einfügen für die Anfangsbuchstaben der Nachnahmen wie im Adressbuch (siehe `UILocalizedIndexedCollation`)
- Wäre ein manuelle Aktualisierung nicht eine gute Idee? Mit einem *Bar Button Item* oder mit einem *Scroll down* wie in der Mail App? (siehe `UIRefreshControl`)
- Sieht die Detailansicht gut und übersichtlich aus? Könnte hier etwas optimiert werden?
- Was passiert eigentlich bei Fehlern? Sollte der Benutzer nicht darüber Informiert werden, wenn das Netzwerk nicht zur Verfügung steht oder der Server gerade nicht reagiert? Wollen wir den Benutzer nicht auch über *In Progress* informieren? (siehe `UIApplication networkActivityIndicatorVisible`)
- Entspricht das User Interface den *Human Interface Guidelines*? (siehe https://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html)
- Die App hat noch kein Icon ... fügen wir doch eines hinzu.
- Internationalisierung ist ein sehr wichtiges Thema für Apps. Können wir die App auch für den Deutschen, Französischen und Italienischen Sprachraum bereit machen?

Fazit

Folgende Punkte haben wir gelernt:

- Einfache *CRUD* (**C**reate, **R**ead, **U**pdate, **D**elete) App zu bauen
-