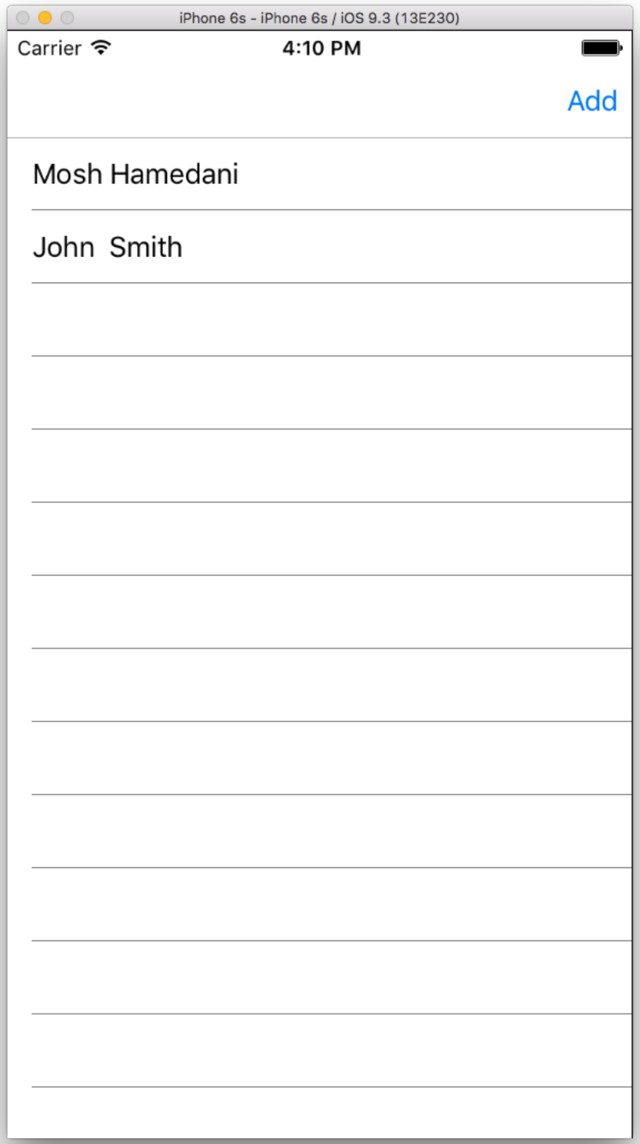
Forms and Setting Pages

By: Mosh Hamedani

ContactBook App

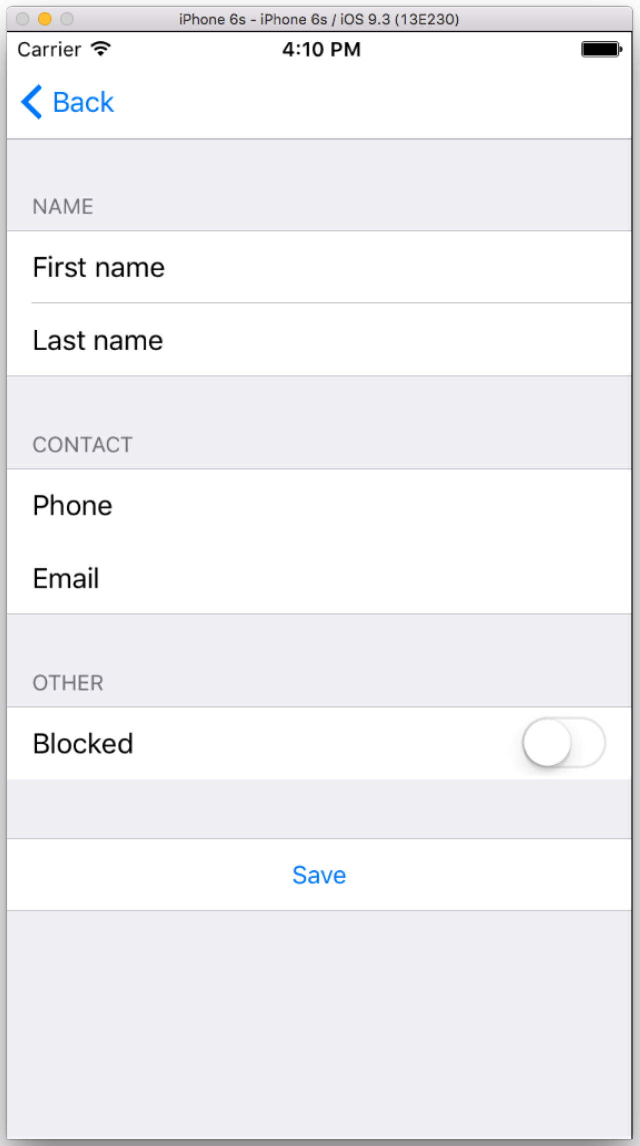
ContactBook is a simple app for managing contacts. This app supports all CRUD operations but changes are not persisted. So, if we quit the application and re-launch it, all changes are lost. In the next section (Data Access) you'll learn how to store these contacts in a SQLite database.



When we click the **Add** button on the toolbar, we go to **ContactDetailPage**.

Forms and Setting Pages

By: Mosh Hamedani



Similarly, when we select an existing contact, we navigate to **ContactDetailPage**. Contact form should be populated with details of the selected contact.

Selected item should be de-selected so when we get back to the **Contacts** page, we can select it again.

For the **Phone** and **Email** fields, we should have customized keyboards.

If we tap the Back button, all changes made to the new or existing contact should be lost.

When we tap the Save button, if both the first and last name fields are empty, we should get an alert with the message: “Please enter the name.” Otherwise, we go back to the **Contacts** page.

Changes made to an existing contact are not visible in the **Contacts** page. In other words, if we change the name of an existing contact and save the changes, we will not see the new name in the list of contacts. However, if we select the contact and go to the **ContactDetailPage** again, the updated name will be there. This is expected behaviour of this application. In the next section, you’ll learn how to solve this problem using **INotifyPropertyChanged** interface. So, for this exercise, don’t worry about it.

In **ContactsPage**, when we swipe a contact to the left, we should see the Delete context action. Tapping that should display an alert to confirm the deletion.

Forms and Setting Pages

By: Mosh Hamedani

Tips

- Use events to communicate between these two pages. So, when the user taps the Save button in **ContactDetailPage**, you should raise an event to which **ContactsPage** has subscribed.

- The event can be either **ContactAdded** or **ContactUpdated**.

- Both these events should use the **Contact** object as their argument. This way, **Contacts** page can see the new/updated contact and add it to its **ListView** or update it:

```
public event EventHandler<Contact> ContactAdded;
```

- Here is how you need to subscribe to this event in **ContactsPage**:

```
var page = new ContactDetailPage();
page.ContactAdded += (source, contact) =>
{
    // do something
};
await Navigation.PushAsync(page);
```

- Use the **Id** of a **Contact** to determine if the contact being edited is a new or an existing contact. If **Id** is 0, set it to 1.

  In a real-world app, **Id** is generated by the database engine. But since we're not using a database, we need to manually set it to a number other than 0.