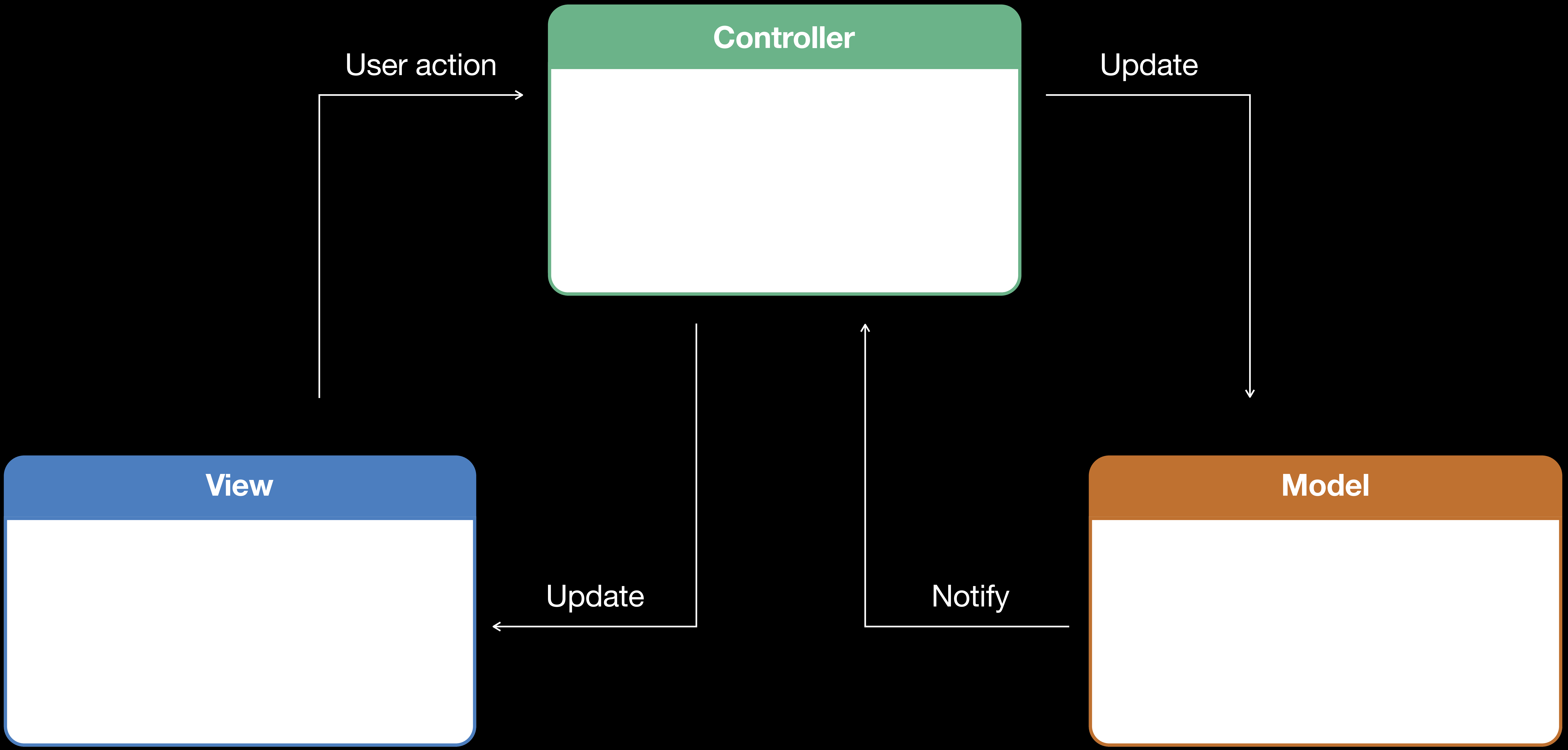
Unit 4—Lesson 7:
Saving Data

Saving data



Encoding and decoding with Codable

```
class Note: Codable {…}
```

Use an Encoder object to encode

Use a Decoder object to decode

Encoding and decoding with Codable

Encoding

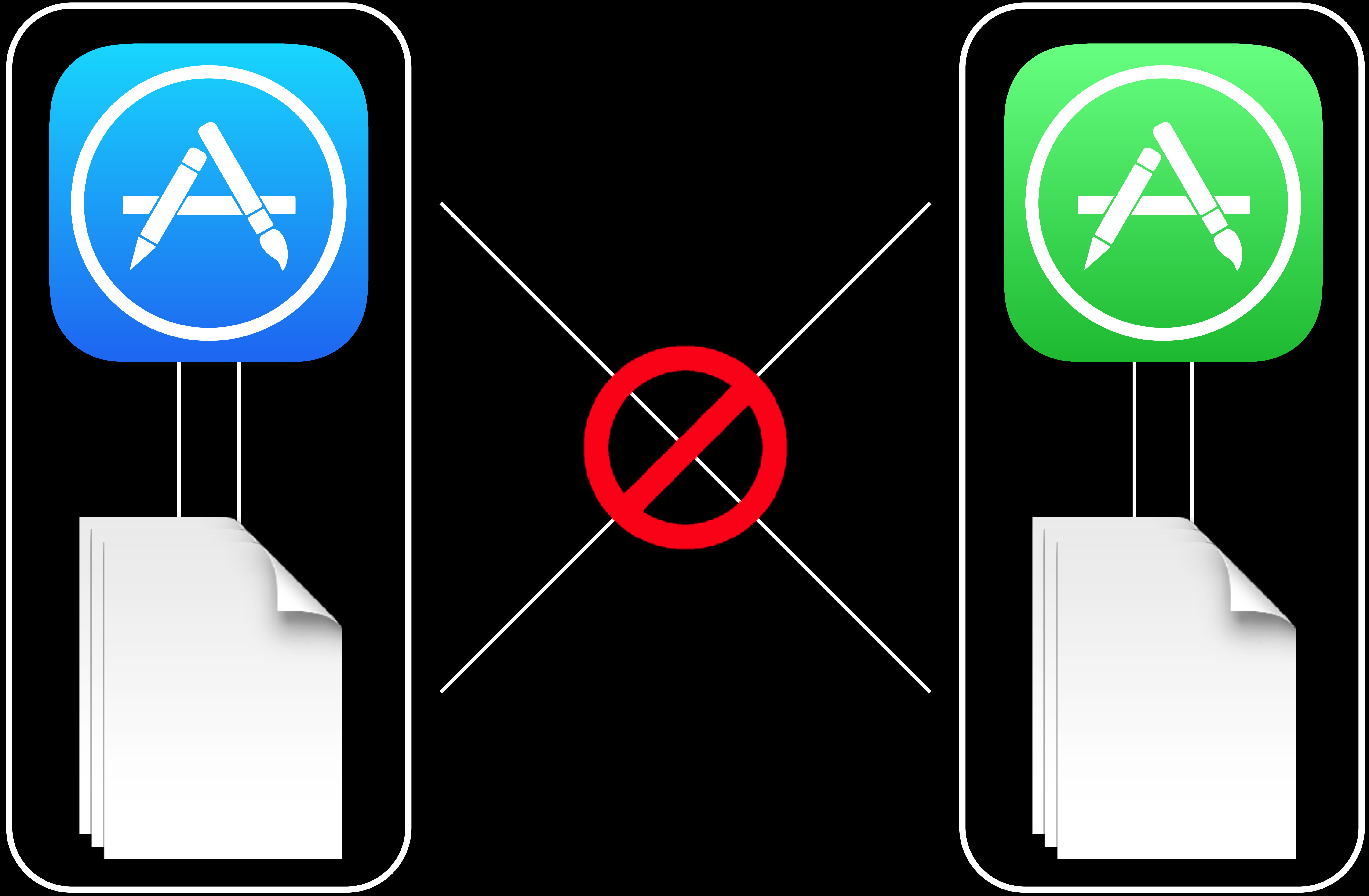
```swift
struct Note: Codable {
    let title: String
    let text: String
    let timestamp: Date
}

let newNote = Note(title: "Dry cleaning", text: "Pick up suit from dry cleaners",
                   timestamp: Date())

let propertyListEncoder = PropertyListEncoder()
if let encodedNotes = try? propertyListEncoder.encode(newNote) {
   . . .
}
```

Encoding and decoding with Codable

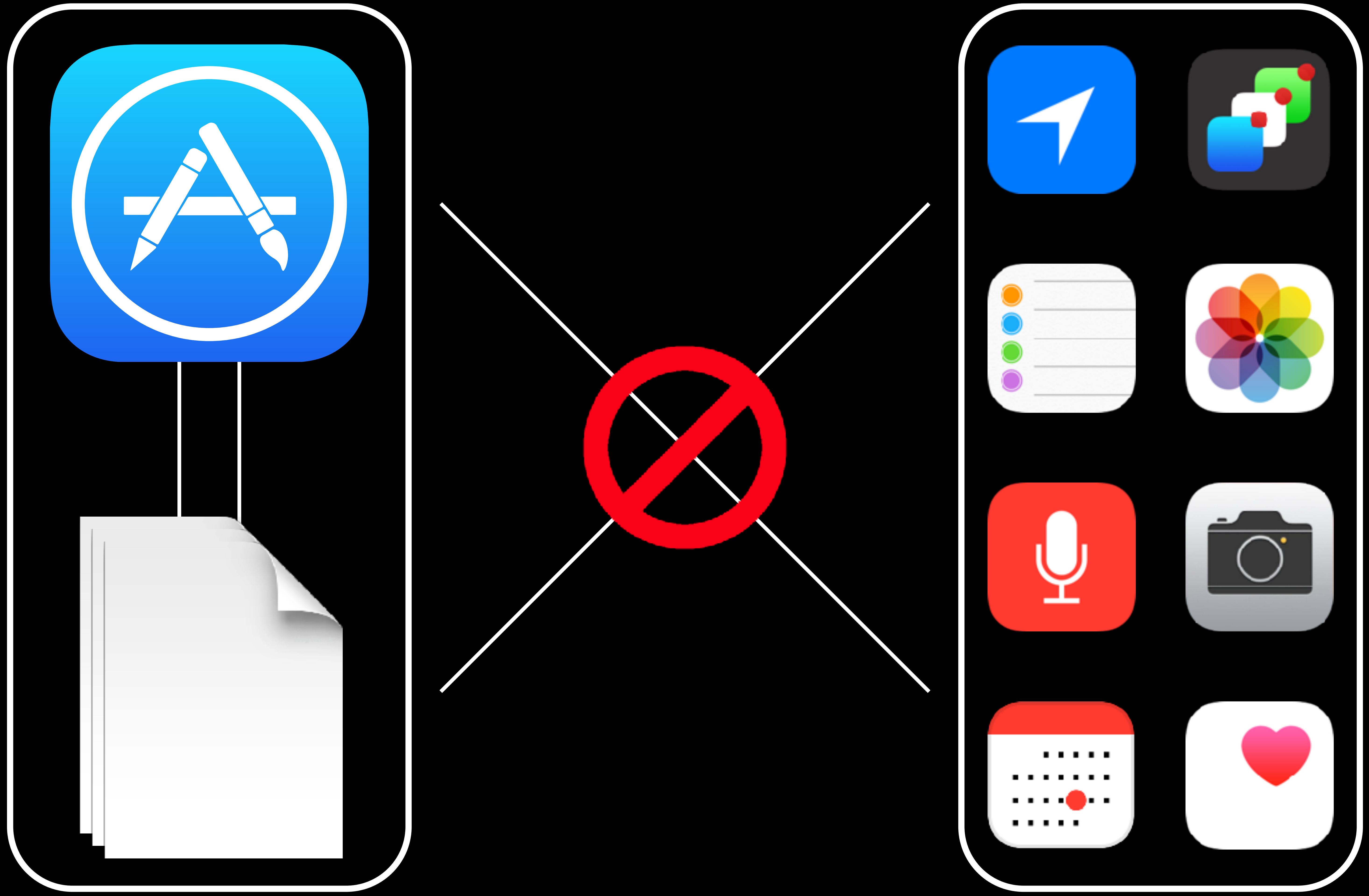
Decoding

```
let propertyListDecoder = PropertyListDecoder()
if let decodedNote = try? propertyListDecoder.decode(Note.self, from: encodedNote) {
   . . .
}
```

App sandbox

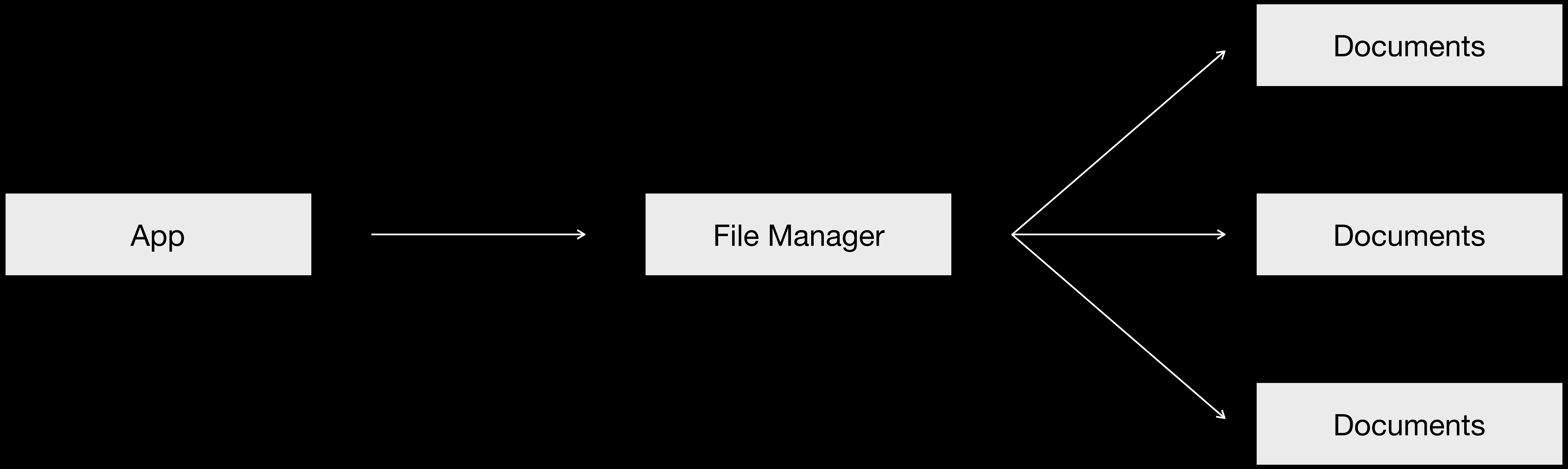


App sandbox



Writing data to a file

Sandboxing



Writing data to a file

Documents directory

```
let documentsDirectory = FileManager.default.urls(for: .documentDirectory,
    in: .userDomainMask).first!
let archiveURL =
    documentsDirectory.appendingPathComponent("appData").appendingPathExtension("plist")
```

Writing data to a file

Writing the data

```
let documentsDirectory = FileManager.default.urls(for: .documentDirectory,
    in: .userDomainMask).first!
let archiveURL =
    documentsDirectory.appendingPathComponent("notes_data").appendingPathExtension("plist")

let propertyListEncoder = PropertyListEncoder()
let encodedData = try? propertyListEncoder.encode(data)

try? encodedData?.write(to: archiveURL, options: .noFileProtection)
```

Writing data to a file

Reading the data

```
let documentsDirectory = FileManager.default.urls(for: .documentDirectory,
    in: .userDomainMask).first!
let archiveURL =
    documentsDirectory.appendingPathComponent("appData").appendingPathExtension("plist")

let propertyListDecoder = PropertyListDecoder()
if let retrievedData = try? Data(contentsOf: archiveURL),
    let decodedNote = try? propertyListDecoder.decode(Note.self, from: retrievedNoteData) {
. . .
}
```

Writing data to a file
Saving an array of model data

```swift
let notes = [note1, note2, note3]

let documentsDirectory = FileManager.default.urls(for: .documentDirectory,
    in: .userDomainMask).first!
let archiveURL =
    documentsDirectory.appendingPathComponent("notes_data").appendingPathExtension("plist")

let propertyListEncoder = PropertyListEncoder()
let encodedData = try? propertyListEncoder.encode(notes)

try? encodedData?.write(to: archiveURL, options: .noFileProtection)
```

Writing data to a file

Reading an array of model data

```
let documentsDirectory = FileManager.default.urls(for: .documentDirectory,
    in: .userDomainMask).first!
let archiveURL =
    documentsDirectory.appendingPathComponent("notes_data").appendingPathExtension("plist")

let propertyListDecoder = PropertyListDecoder()
if let retrievedNotesData = try? Data(contentsOf: archiveURL),
   let decodedNotes = try? propertyListDecoder.decode(Array<Note>.self,
                                                       from: retrievedNotesData) {
    ...
}
```

Remember

Your model objects should implement the `Codable` protocol.

Reading and writing should happen in the model controller.

Archive in the correct app delegate life-cycle events. For example:

- When the app enters the background
- When the app is terminated

Unit 4—Lesson 7

Saving Data

Learn how to persist data using `Codable`, a protocol for saving files to your app's Documents directory.

Unit 4—Lesson 7

Lab: Remember Your Best Friends

Use the `Codable` protocol to persist information between launches of an app listing your best friends.