SCIENCES SORBONNE UNIVERSITÉ

# Using contacts

Fabrice.Kordon@lip6.fr

LIP6

# As an introduction…



## Phone + Contact book

- Key elements in the system
- Significant added value to your applications


## A bit of history

- An old framework AddressBookUI
  - **Mix of C& Objective-C**
- New framework since iOS9
  - **The one we study of course**

# ContactsUI & Contact



## New frameworks…

- CNContact / CNMutableContact
  - Description of a contact
- CNContactStore
  - Search, access, modification of contacts
  - Bach save thanks to CNSaveRequest
- CNContactPickerViewController
  - Allows to select one or more contacts
  - Associated to CNContactPickerDelegate protocol
- CNContactViewController
  - Display of a contact
  - Associated to CNContactViewControllerDelegate protocol

## Rich frameworks

- Objective, giving you a global vision on how it works

# CNContact & CNMutableContact



## Describe a contact

- CNContact is thread safe
- CNMutableContact is not
- Transformation
  - **mutableCopy(with:)**

## Main atributes

- Identifier (unique for a given device), String
- contactType (CNContactType = person, organization)
- namePrefix, givenName, middleName, familyName, previousFamilyName, nameSuffix, nickName, jobTitle, etc.
  - **All Strings**
- phoneNumber, postalAdresses, emailAdresses, etc.
  - **Arrays of CNLabeledValues (generic class)**
- imageData, thumbnailImageData

# CNLabeledValues



- **Key / Value**
  - Key of type String
  - Value of any type
- **Much used in CNContacts**
  - CNPhoneNumber
    - **stringValue property**
  - CNPostalAddress
    - **Properties: street, city, state, postalCode, country, isoCountryCode, etc**
    - **Method localizedString(forKey:)**
  - And many more in the RTFM!

# CNContactStore



## Load, update and save contacts

- Authorisations required
  - requestAccess(for:completionHandler:)
  - Handler executed in case of success
- Unified contacts
  - Fusioning several contacts

## Read mode

- Fetching elements from the contact book
  - Possible use of predicates to build queries (CNContactFetchRequest)

## Write mode

- Contacts asynchronously saved
  - Use of another thread

## And even more

- Have a look at the

# CNContactPickerViewController



## Handles selection of contacts

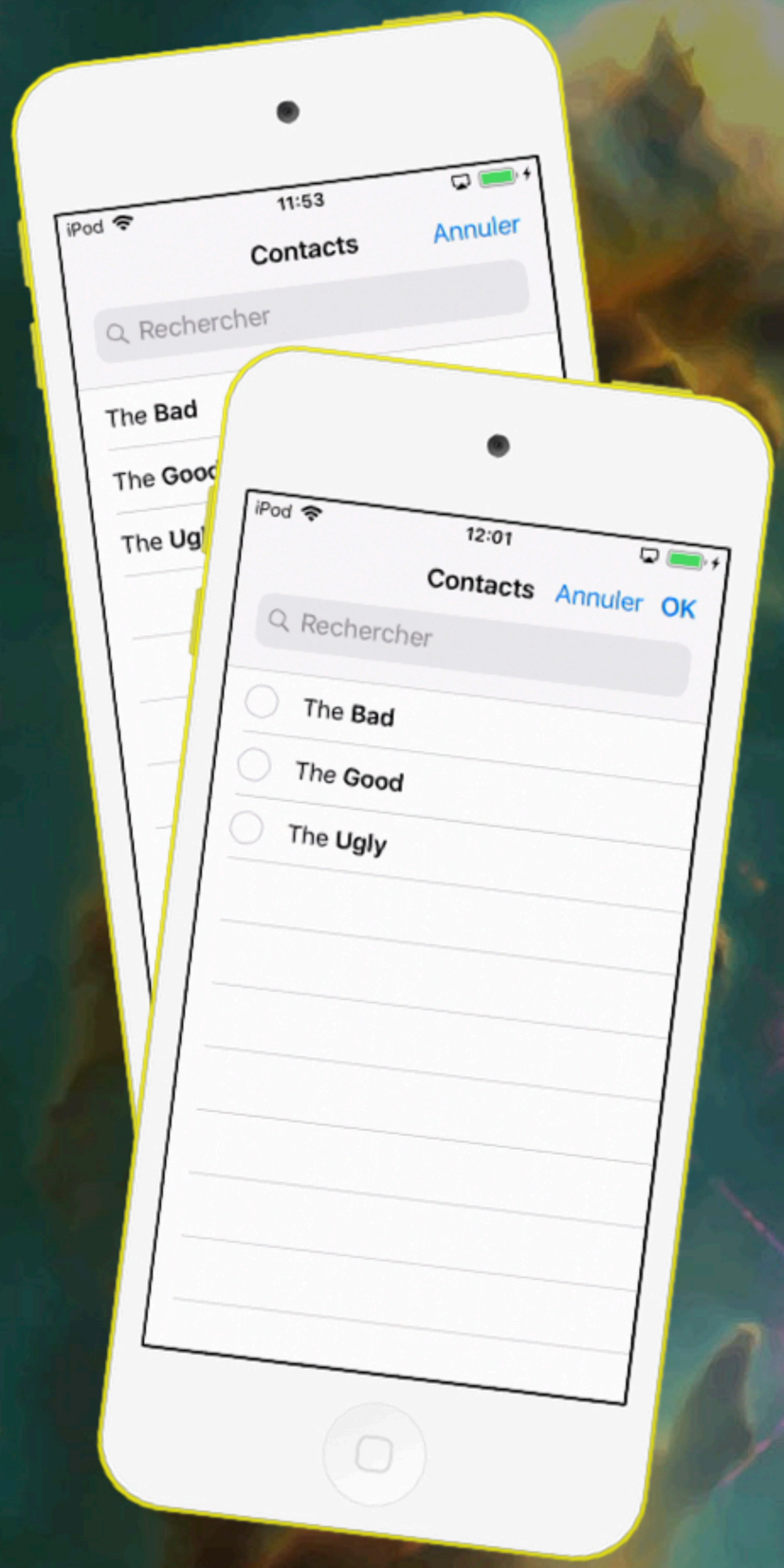
- One or several

## Principle

- Presented by a view controller
- Support of CNContactPickerDelegate
  - contactPickerDidCancel(_:)
  - contactPicker(_:didSelect:)
    Returns a CNContact or a [CNContact]
    Chose the one you want
  - contactPicker(_:didSelectContactProperties:)
    To let people select properties in contacts
    Returns a [CNContactProperty]

# CNContactPickerViewController





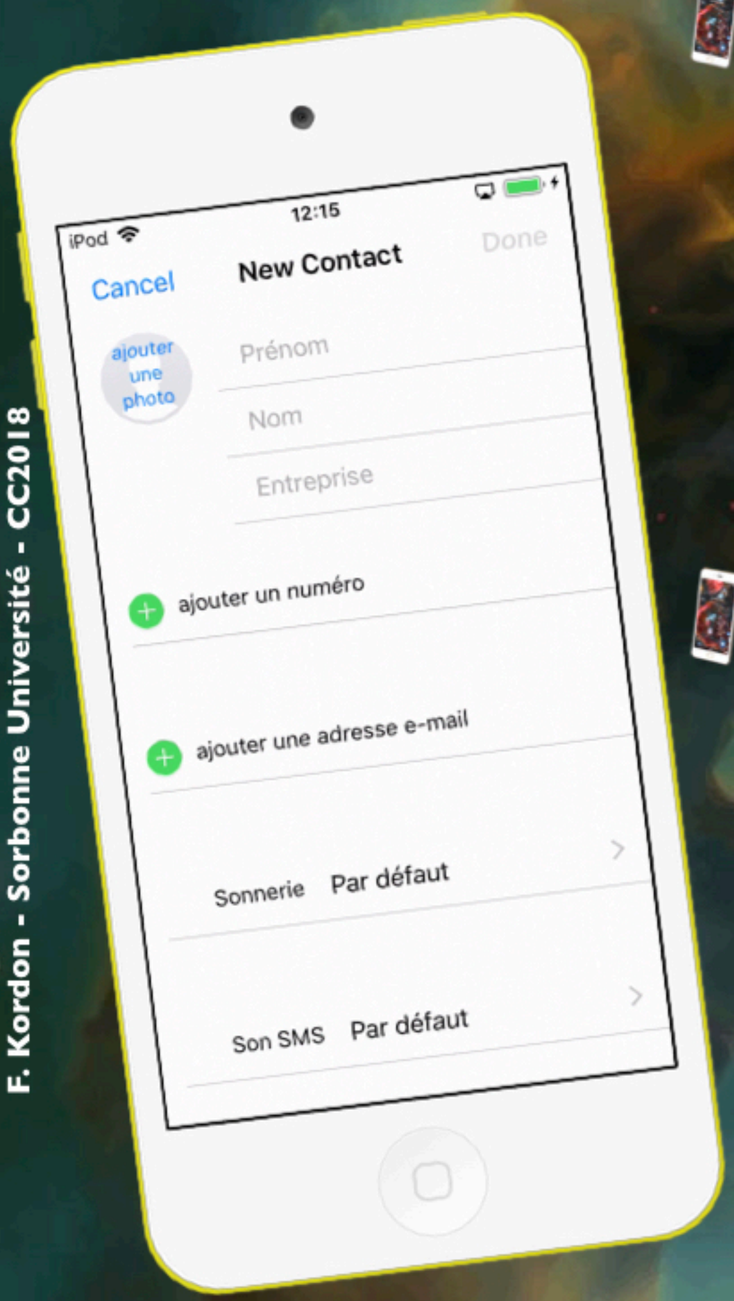

## Handles selection of contacts

- One or several

## Principle

- Presented by a view controller
- Support of CNContactPickerDelegate
  - contactPickerDidCancel(_:)
  - contactPicker(_:didSelect:)
    Returns a CNContact or a [CNContact]
    Chose the one you want
  - contactPicker(_:didSelectContactProperties:)
    To let people select properties in contacts
    Returns a [CNContactProperty]

# CNContactViewController




## Presents a contact

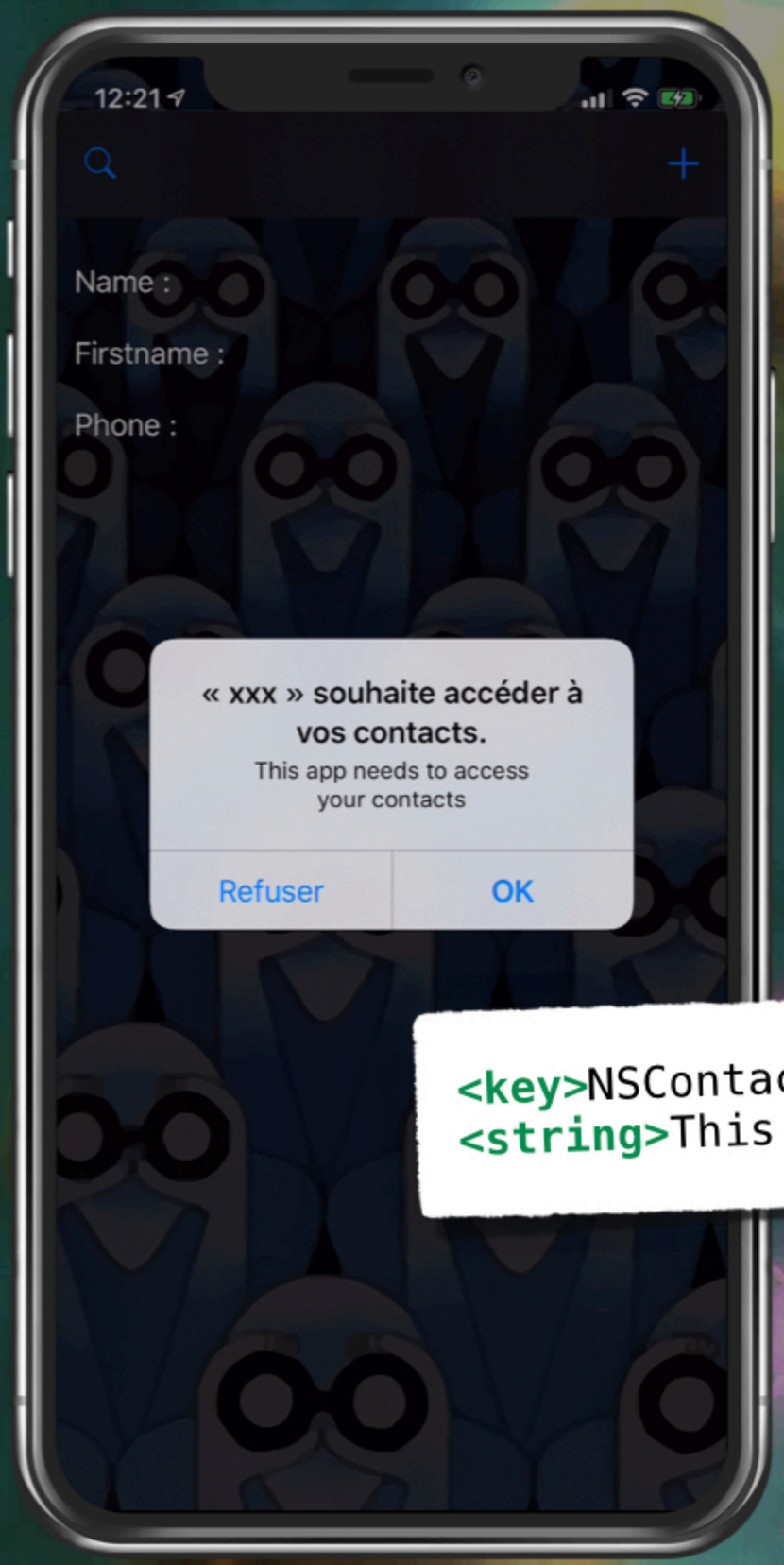
- Existing one of new one depending on init
  - **init(for:)**
  - **init(forUnknownContact:)**
  - **init(forNewContact:)**

## Principle

- Presented by a view controller
- Support ofCNContactViewControllerDelegate
  - **contactViewController(_:didCompleteWith:)**
    **called when he view has just been presented**
  - **contactViewController(_:shouldPerformDefaultActionFor:)**
    **called when a property is selected**

# Privacy concerns






- **You must declare access in the info.plist**
  - Key NSContactsUsageDescription

```
<key>NSContactsUsageDescription</key>
<string>This app needs to access your contacts</string>
```

# As a conclusion…



## It is a complex framework

- But it is far easier than the previous one

## Remember to check for access

- With a CNContactStore
  - **handler executed when access granted**
- Declaration in the info.plist too
  - **NSContactsUsageDescription**