範例analytic
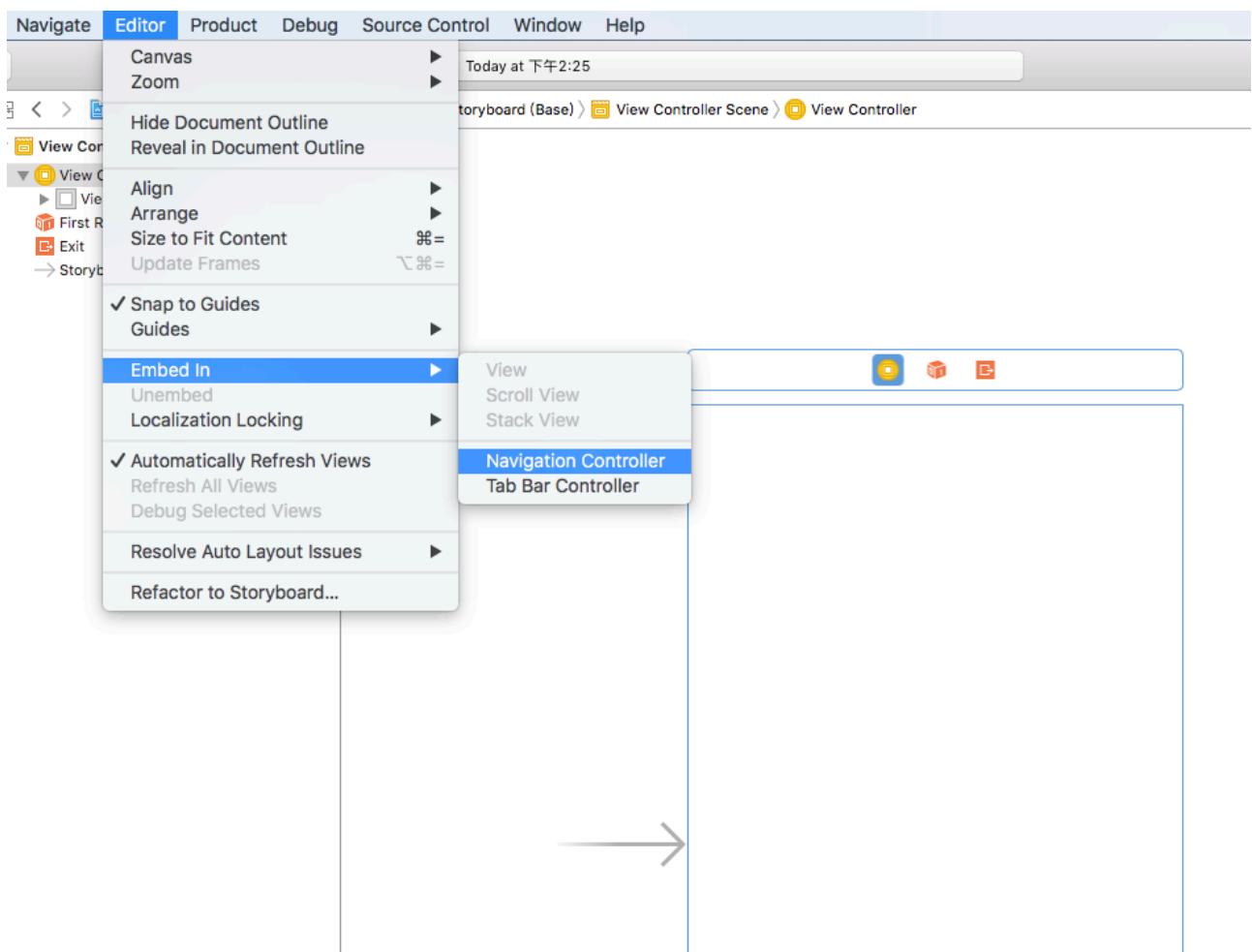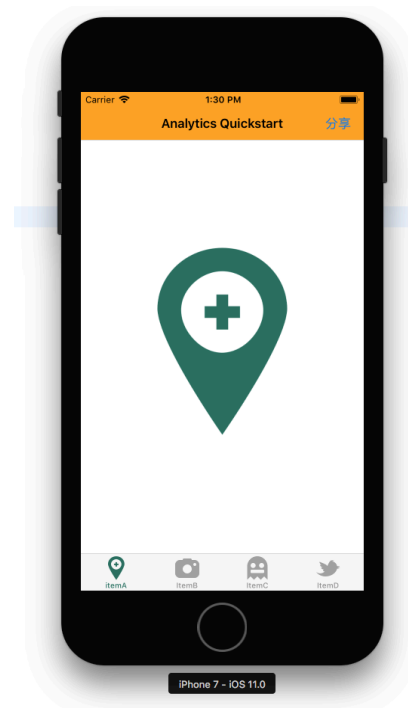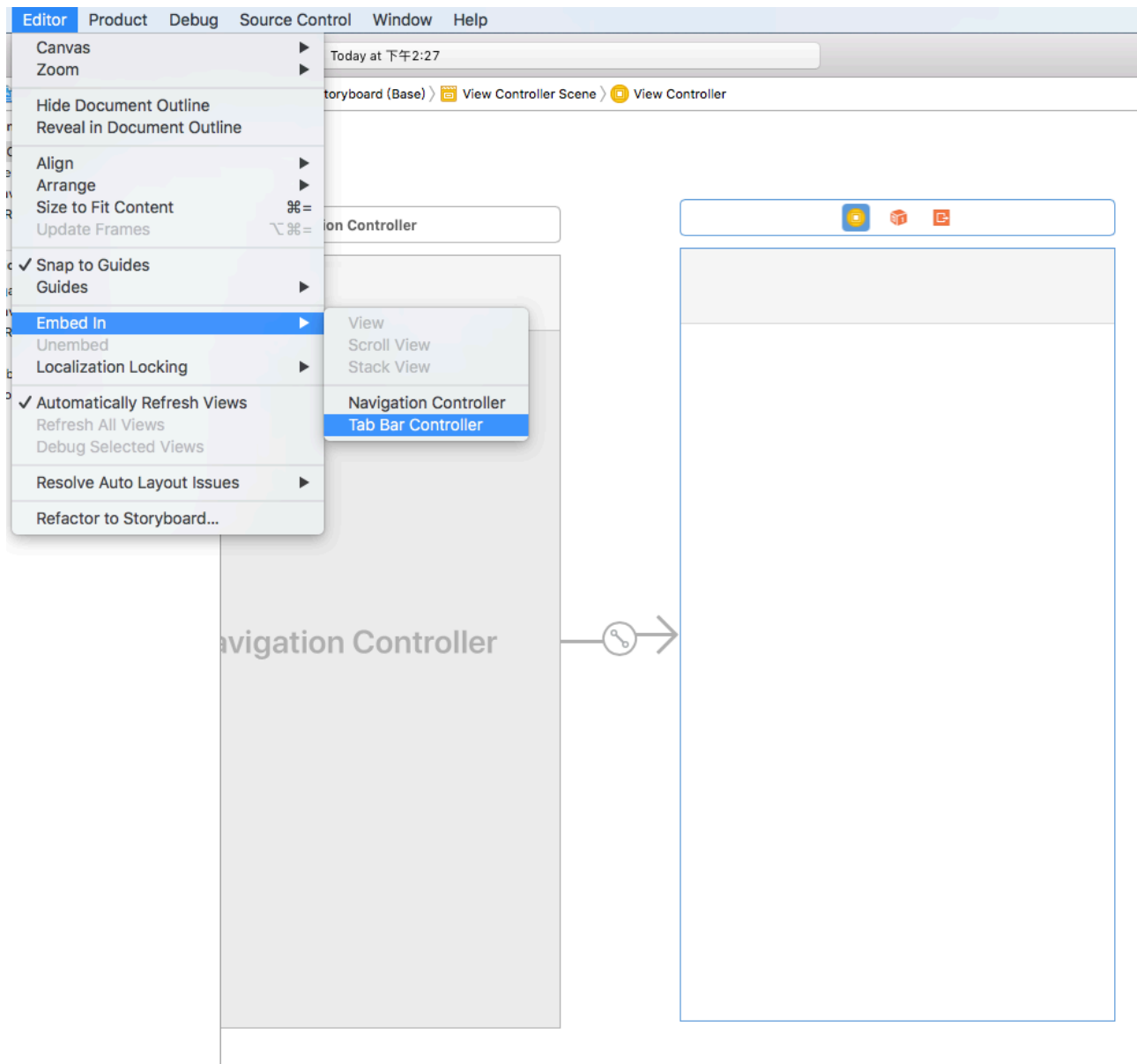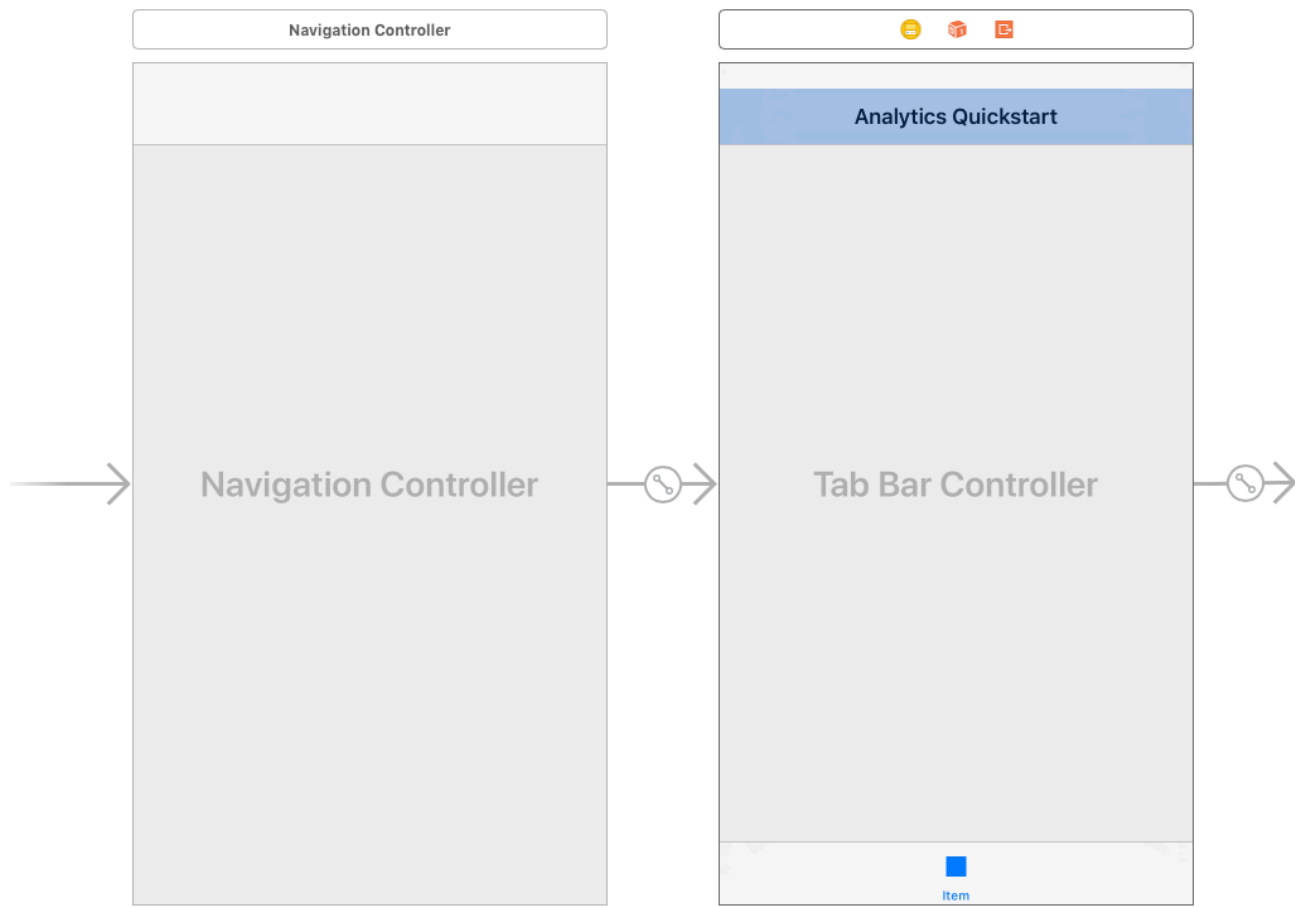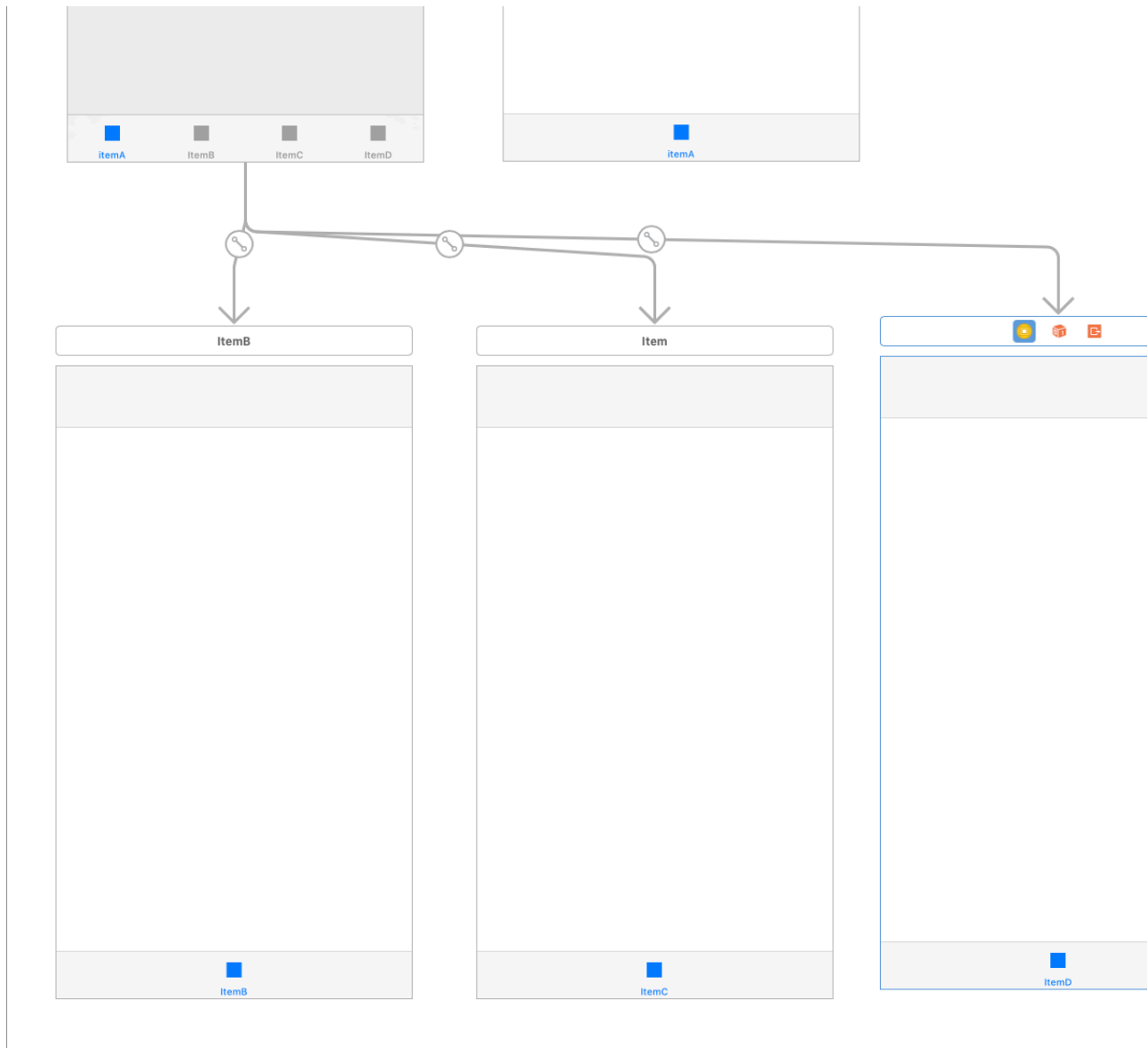
將UINavigationConroller包圍ViewController。

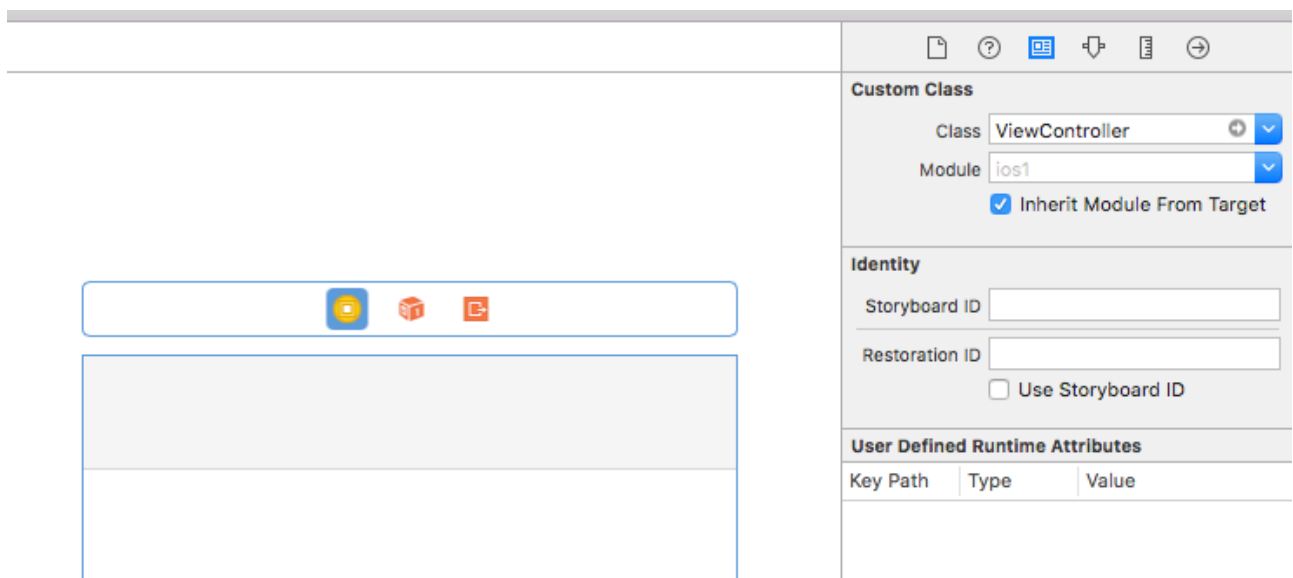再將UITabBarController包圍ViewConroller

將UITabBarController的NavigationItem的title，改為Analytics Quickstart。

將UITabBarController加入另外3個UIViewController，並將UITabBarItem分別變為 ItemA,itemB,itemC,itemD。

將4頁的UIViewController，全部皆改為ViewController。

將UIBarButtonIem放在TabBarConroller最上方，並將BarItem改為分享。
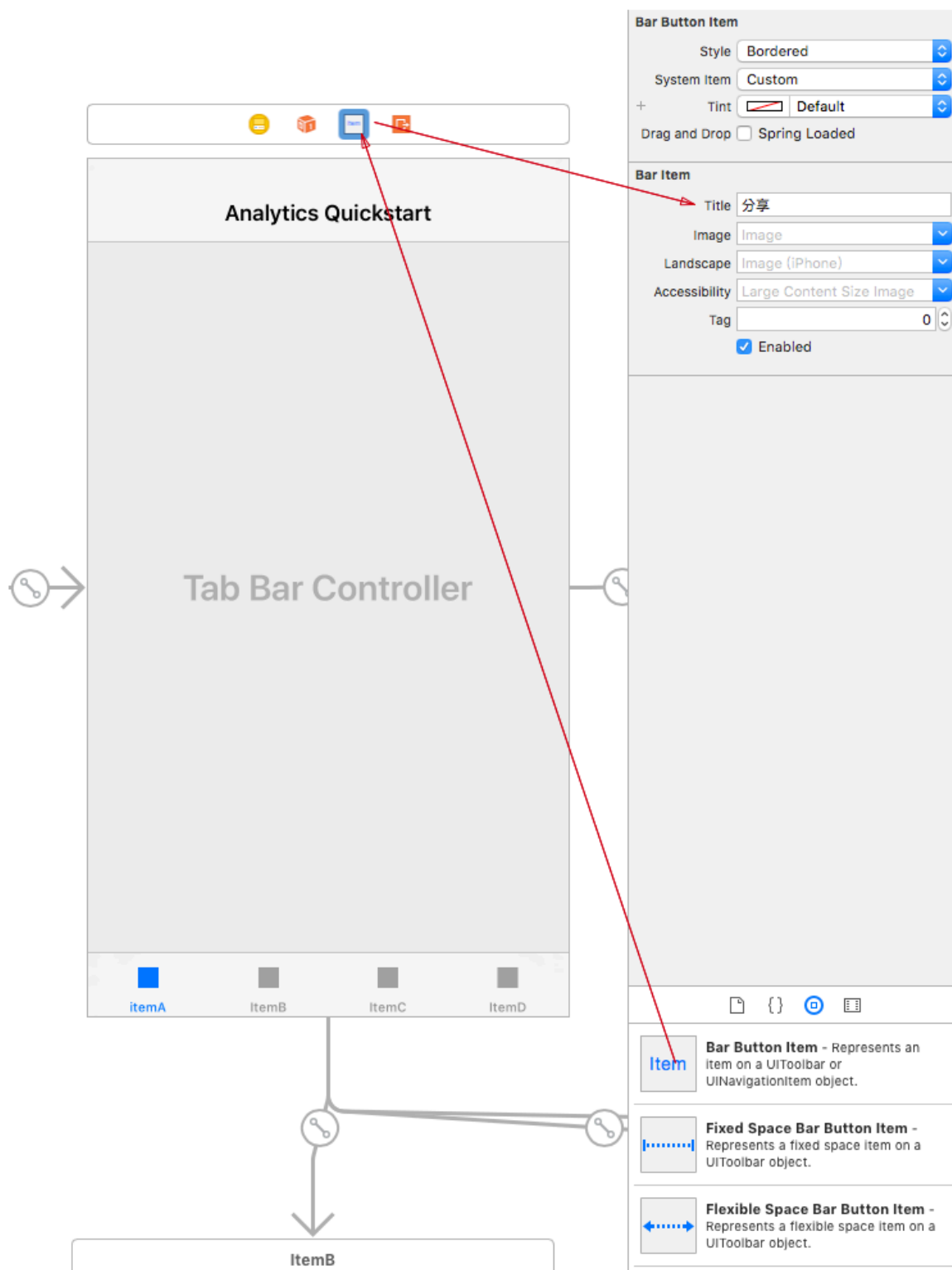
將navigationItem的rightBarButtomItem,連結至分享的barButtonItem。
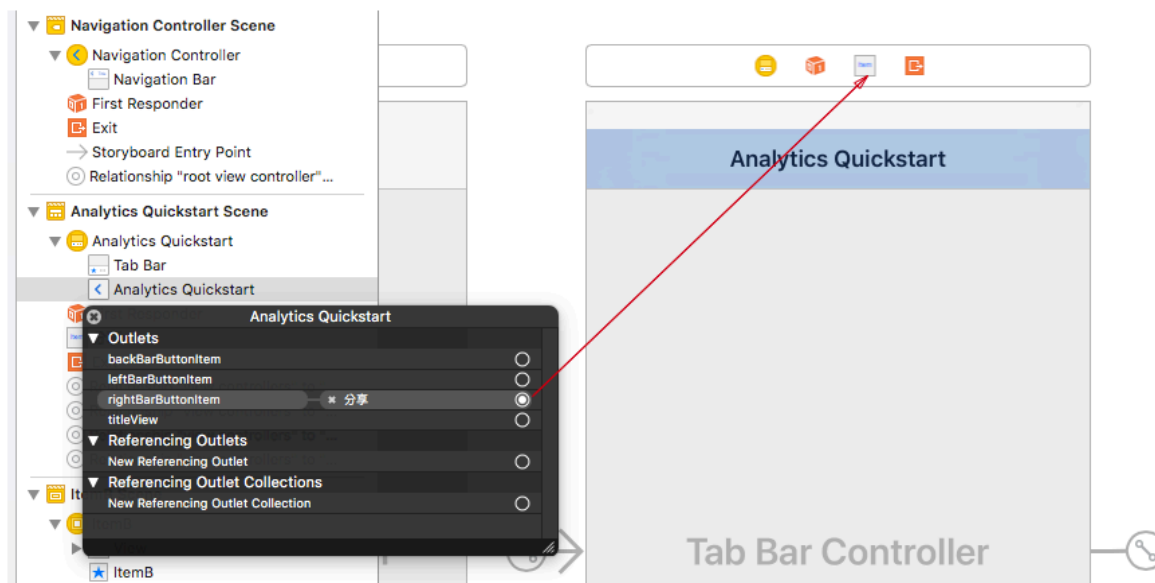
將AppDelegate加入下列程式碼,初始化Firebase

```swift
1
2  import UIKit
3  import Firebase
4
5  @UIApplicationMain
6  class AppDelegate: UIResponder, UIApplicationDelegate {
7
8      var window: UIWindow?
9
10
11     func application(_ application: UIApplication,
           didFinishLaunchingWithOptions launchOptions:
           [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
12         // Override point for customization after application launch.
13         FirebaseApp.configure();
14
15         return true
16     }
17
18
19
20
21 }
22
23
```

在storyboard建立新的空白的UIViewController，並在專案新增
FoodPickerViewController，繼承UIViewController。將storyboard內空的
UIViewConroller的custom class改為FoodPickerViewController。

Choose options for your new file:

Class: FoodPickerViewConroller

Subclass of: UIViewController

☐ Also create XIB file

Language: Swift

⚠ 1

🔲 Food Picker View Conroller   < ⚠ >

Custom Class

Class   FoodPickerViewConroller

Module   ios1

☑ Inherit Module From Target

Identity

Storyboard ID

Restoration ID

☐ Use Storyboard ID

User Defined Runtime Attributes

Key Path   Type   Value

＋ －

Document

Label   Xcode Specific Label

Object ID   0DU-vp-rIS

Lock   Inherited - (Nothing)

Notes

No Font

Comment For Localizer

加入UILabel，將文字改為選擇您喜歡的食物，並且加上Constraints
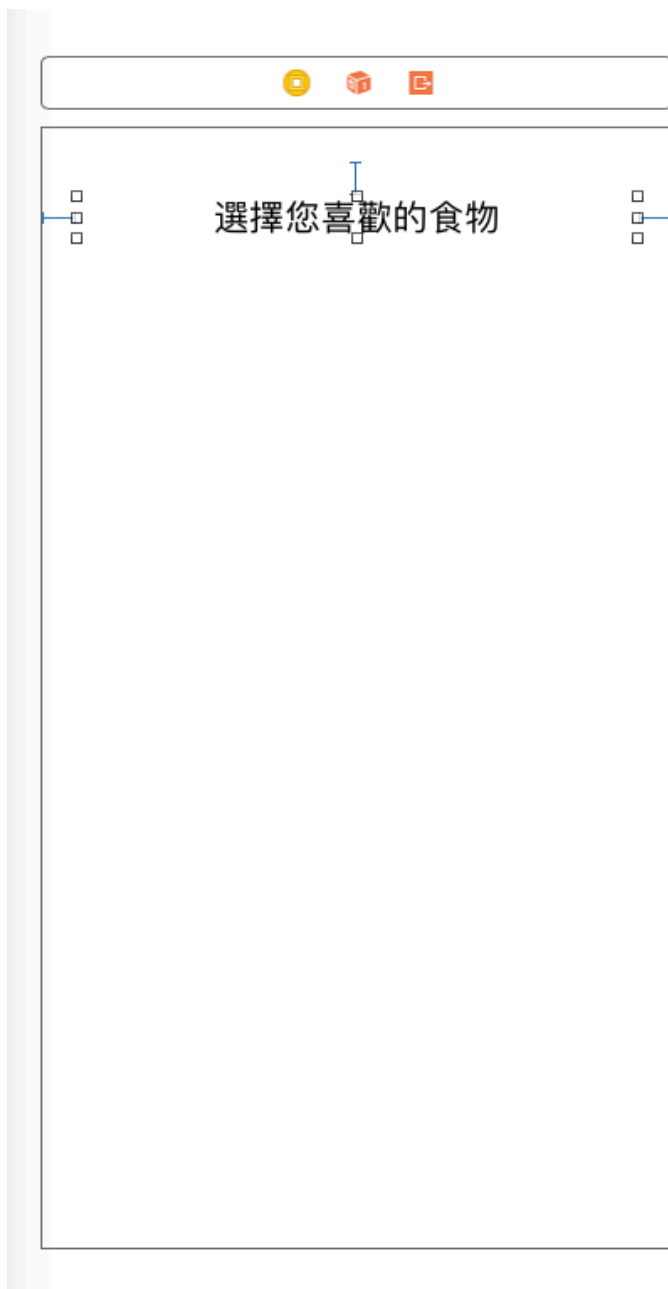
加上UIPickerView，再加上Constraints

選擇您喜歡的食物

Mountain View
Sunnyvale
Cupertino
Santa Clara
San Jose
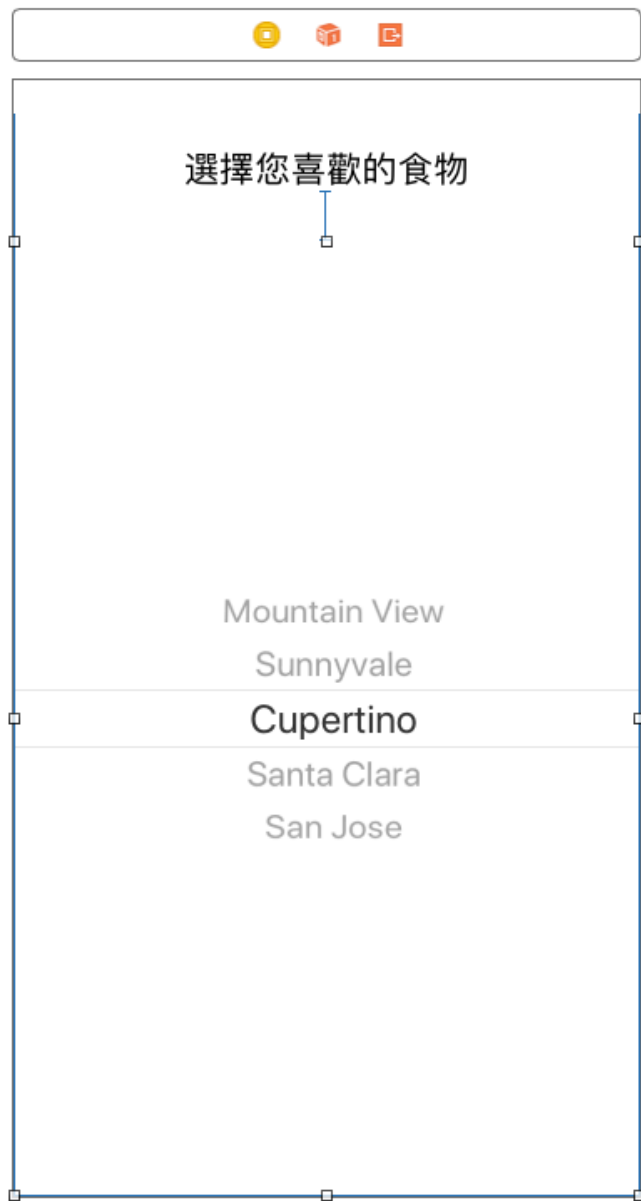
建立PatternTabBarController:UITabBarController，將storyboard的
UITabBarController的custom class改為PatternTabBarController
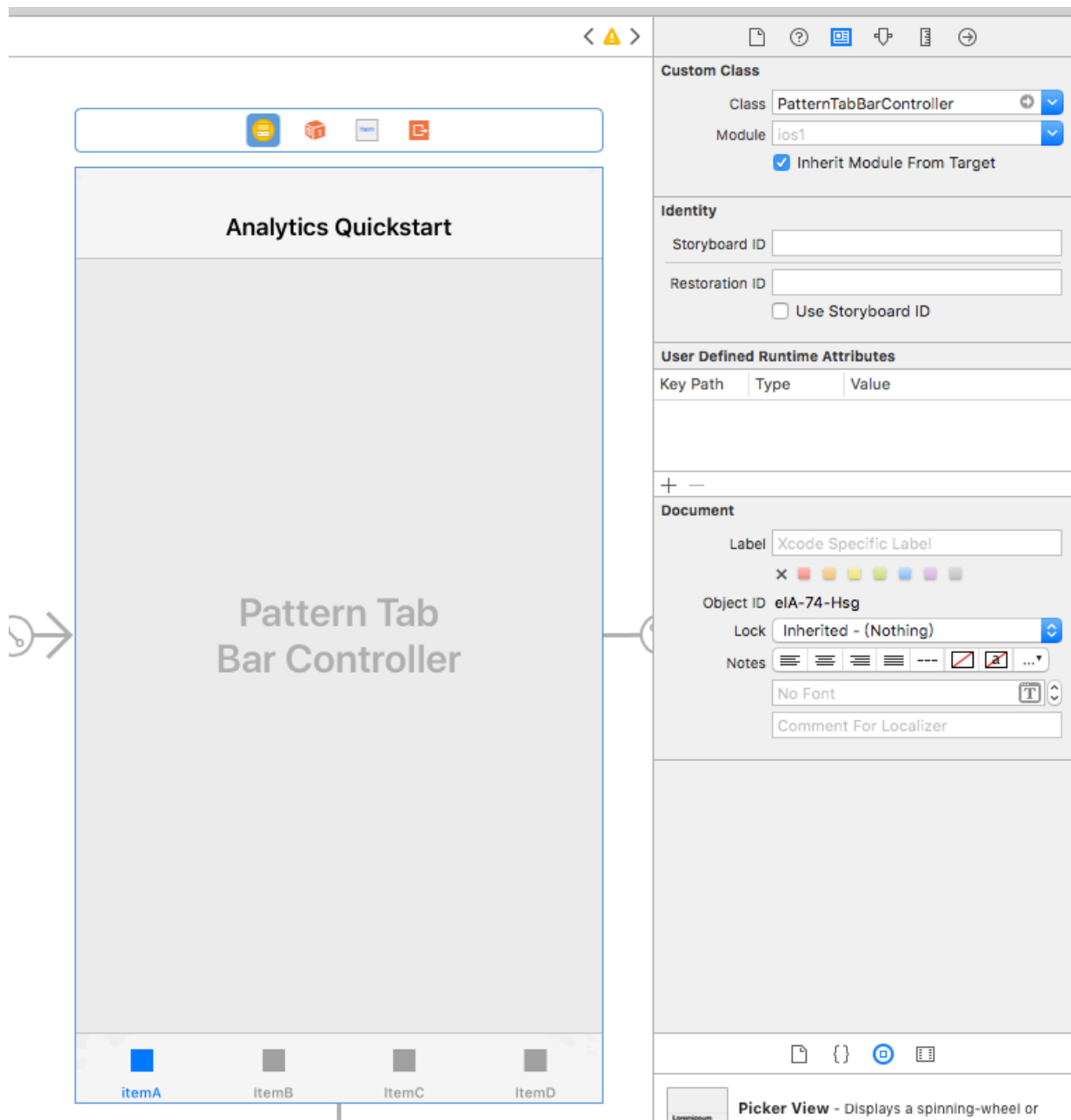
由UITabBarController按住control,並拖滑鼠至FoodPickerViewController，選擇
presend Modally。

將segue identifier設定為pickFavoriteFood
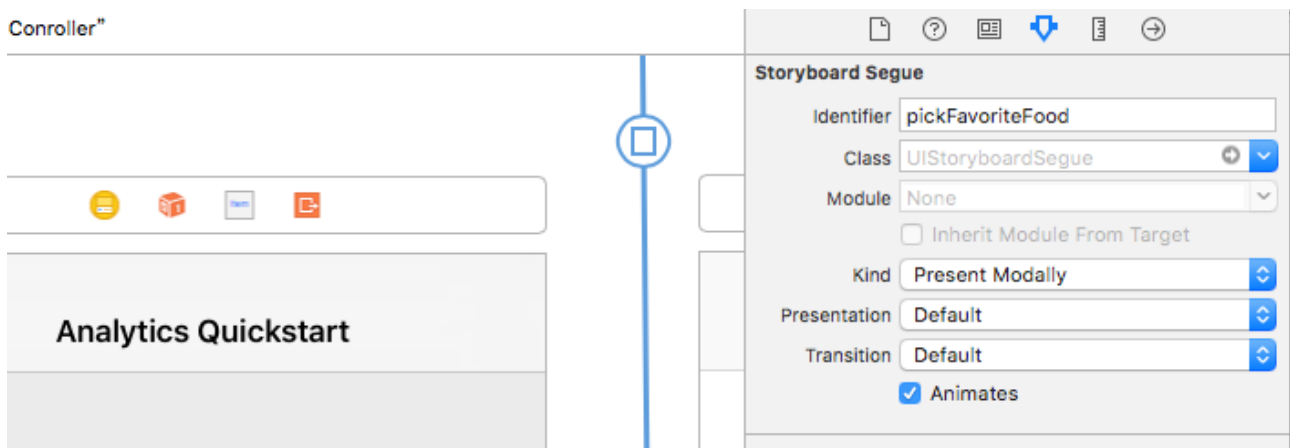
打開PatternTabBarController.swift, 建立 方法getUserFavorieFood()和
askForFavoriteFood()，getUserFavoriteFood()主要功能是檢查App永久儲存的
library內有沒有儲存key為"favorite_food"的值，沒有就傳出nil。askForFavoriteFood
的功能是執行segue，讓顯示出FoodPickerController。

```swift
1
2  import UIKit
3
4  class PatternTabBarController: UITabBarController {
5
6      override func viewDidLoad() {
7          super.viewDidLoad()
8
9          // Do any additional setup after loading the view.
10     }
11
12     func getUserFavorieFood() -> String?{
13         return UserDefaults.standard.value(forKey: "favorite_food") as? String;
14     }
15
16     func askForFavoriteFood(){
17         performSegue(withIdentifier: "pickFavoriteFood", sender: nil);
18     }
19
20
21  }
22  |
```

建立override viewDidAppear(_:)，在此執行檢查的動作，如果沒有favorite_food的
值，則跳出FoodPickerConroller.

```swift
import UIKit

class PatternTabBarController: UITabBarController {

    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated);
        if getUserFavorieFood() == nil {
            askForFavoriteFood();
        }
    }

    func getUserFavorieFood() -> String?{
        return UserDefaults.standard.value(forKey: "favorite_food") as?
            String;
    }

    func askForFavoriteFood(){
        performSegue(withIdentifier: "pickFavoriteFood", sender: nil);
    }

}
```

開始編輯FoodPickerViewConroller，在storyboard內建立pickerView的delegate和
dataSource皆為FoodPickerViewController。
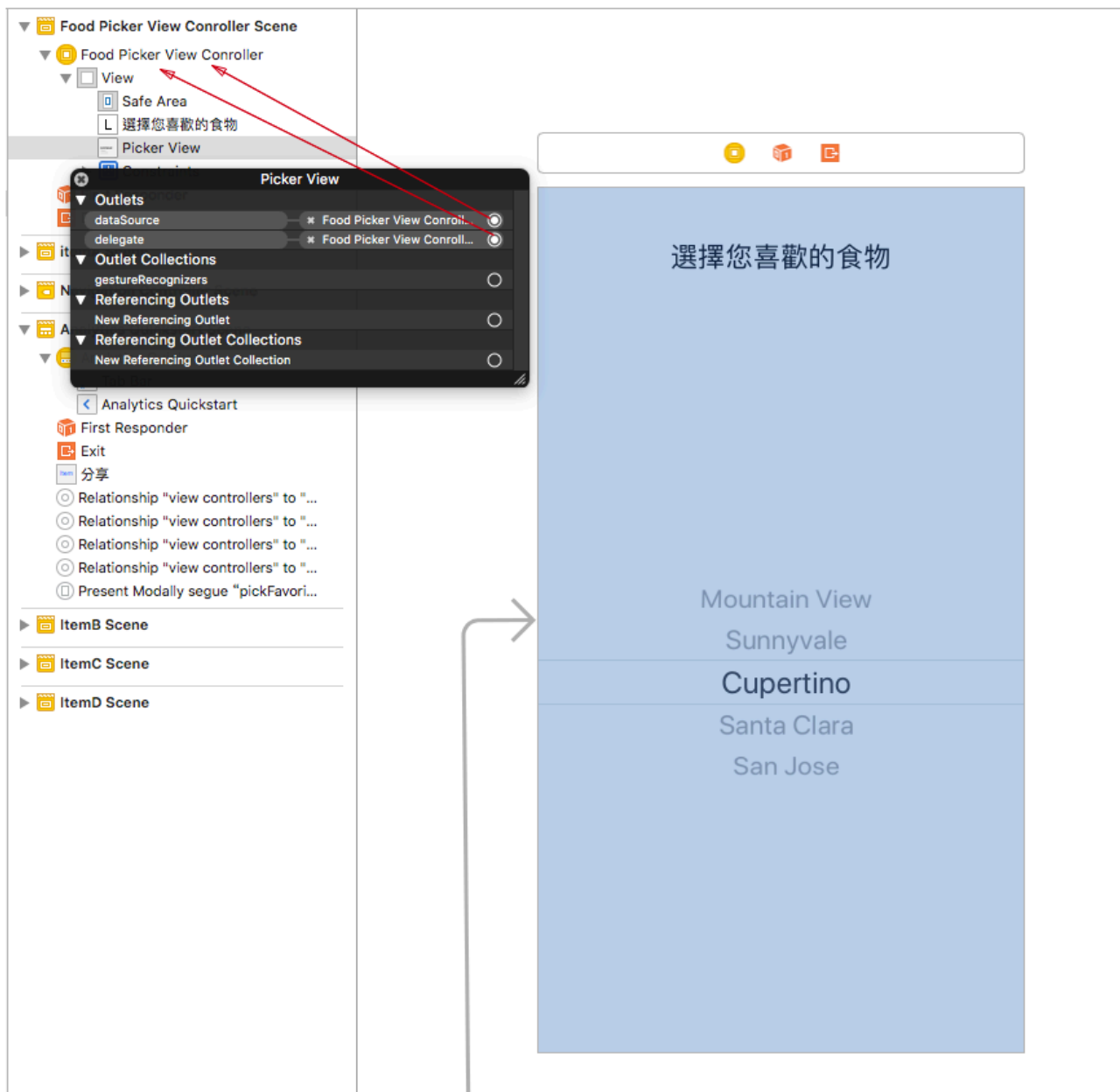
實作FoodPickerViewController。使用extension功能，讓FoodPinckerViewController
採納UIPickerViewDelegate和採納UIPickerViewControllerDataSource。並且實作這2
個protocol內的需求method。

```swift
import UIKit

class FoodPickerViewConroller: UIViewController {
    let foodStuffs = ["Hot Dogs", "Hamburger", "Pizza"];    // 建立顯示內容的陣列

    override func viewDidLoad() {
        super.viewDidLoad()
    }

}

extension FoodPickerViewConroller:UIPickerViewDelegate{    // 實作UIPicerViewDelegate
    func pickerView(_ pickerView: UIPickerView, titleForRow row: Int,
        forComponent component: Int) -> String?{
        return foodStuffs[row];    // 回傳row要顯示的內容
    }

    func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int,
        inComponent component: Int){

    }
}

extension FoodPickerViewConroller:UIPickerViewDataSource{    // 實作UIPicerViewDataSource
    func numberOfComponents(in pickerView: UIPickerView) -> Int {
        return 1;    // 回傳componens數量
    }

    func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent
        component: Int) -> Int {
        return foodStuffs.count;    // 回傳componens內的row的數量
    }

}
```

實作pickerView(_:, didSelectRow:, inComponent:)，使用者選取喜歡吃的食物後，將會執行這個方法。它有2個重要功能
1.儲存使用者選取的選項值至永久儲存區(library)內,且將設為"favorite_food"。
2.傳送user property至Firebase

```swift
import UIKit
import Firebase          ← 載入Firebase

func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int,
    inComponent component: Int){
    let food = foodStuffs[row];
    UserDefaults.standard.set(food, forKey: "favorite_food");     // 儲存資料至library
    UserDefaults.standard.synchronize();

    Analytics.setUserProperty(food, forName: "favorite_food");    // 傳送資料至Firebase
}
```
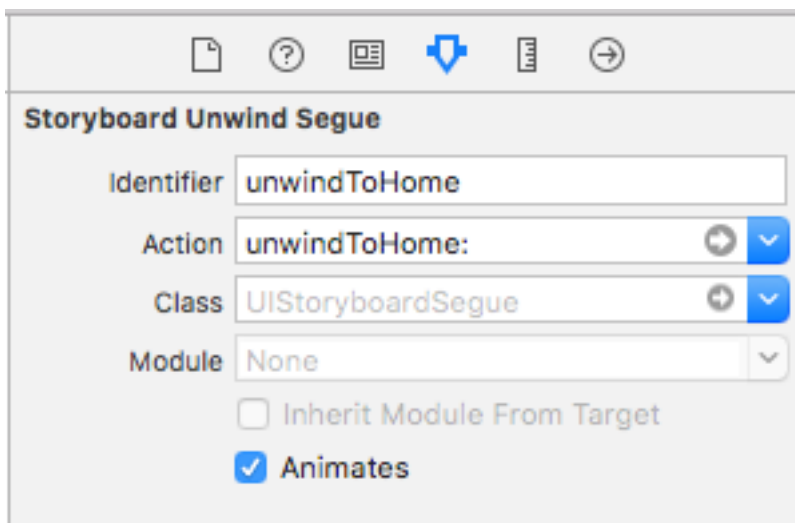
建立退出FoodPickerViewController的segue
1.在PatternTabBarController內建立unwind的方法;

```swift
func getUserFavorieFood() -> String?{
    return UserDefaults.standard.value(forKey: "favorite_food") as? String;
}
```

2在storyboard內, 建立FoodPickerViewConroller 的unwind segue,並命名
為"unwindToHome"。

3.在FoodPickerViewController執行unwind segue。

```swift
func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int,
    inComponent component: Int){
    let food = foodStuffs[row];
    UserDefaults.standard.set(food, forKey: "favorite_food");
    UserDefaults.standard.synchronize();

    Analytics.setUserProperty(food, forName: "favorite_food");

    performSegue(withIdentifier: "unwindToHome", sender: nil);
}
```
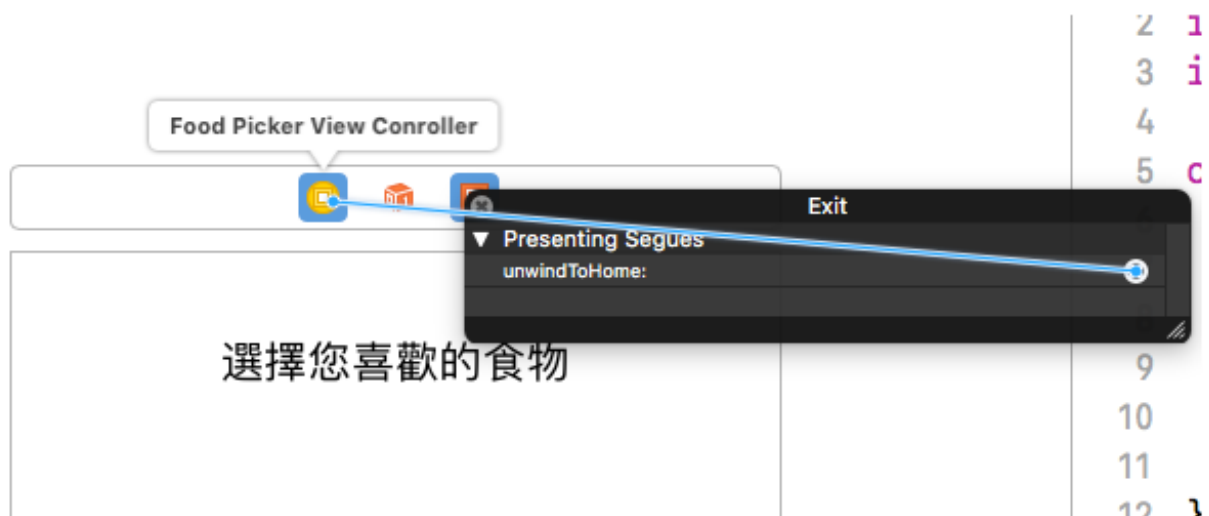
在Firebase console內註冊user property(favorite_food):
1.進入Firebase console 的 Analytics
2.點選UserProperties
3.點選NEW USER PPROPERTY
4.在User property name欄位加入favorite_food
5.在Description欄位加入說明
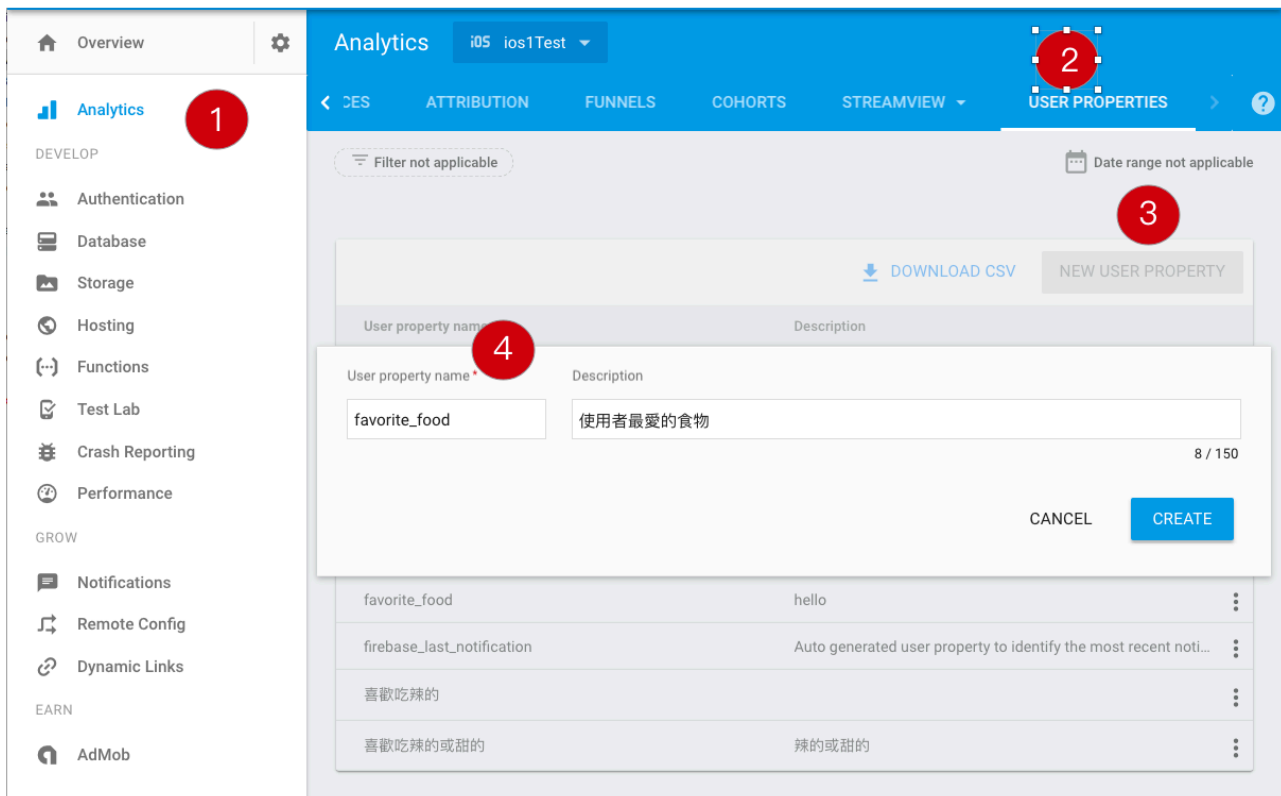
在Xcode debug 視窗看到這個user property ,打開 Analytics debugging:

1. 在xcode選擇 **Product > Scheme > Edit scheme...**

2. 在左邊menu選擇Run

3. 選擇 **Arguments** 的標籤

4. 在 **Arguments Passed On Launch** 加入 `-FIRAnalyticsDebugEnabled.`
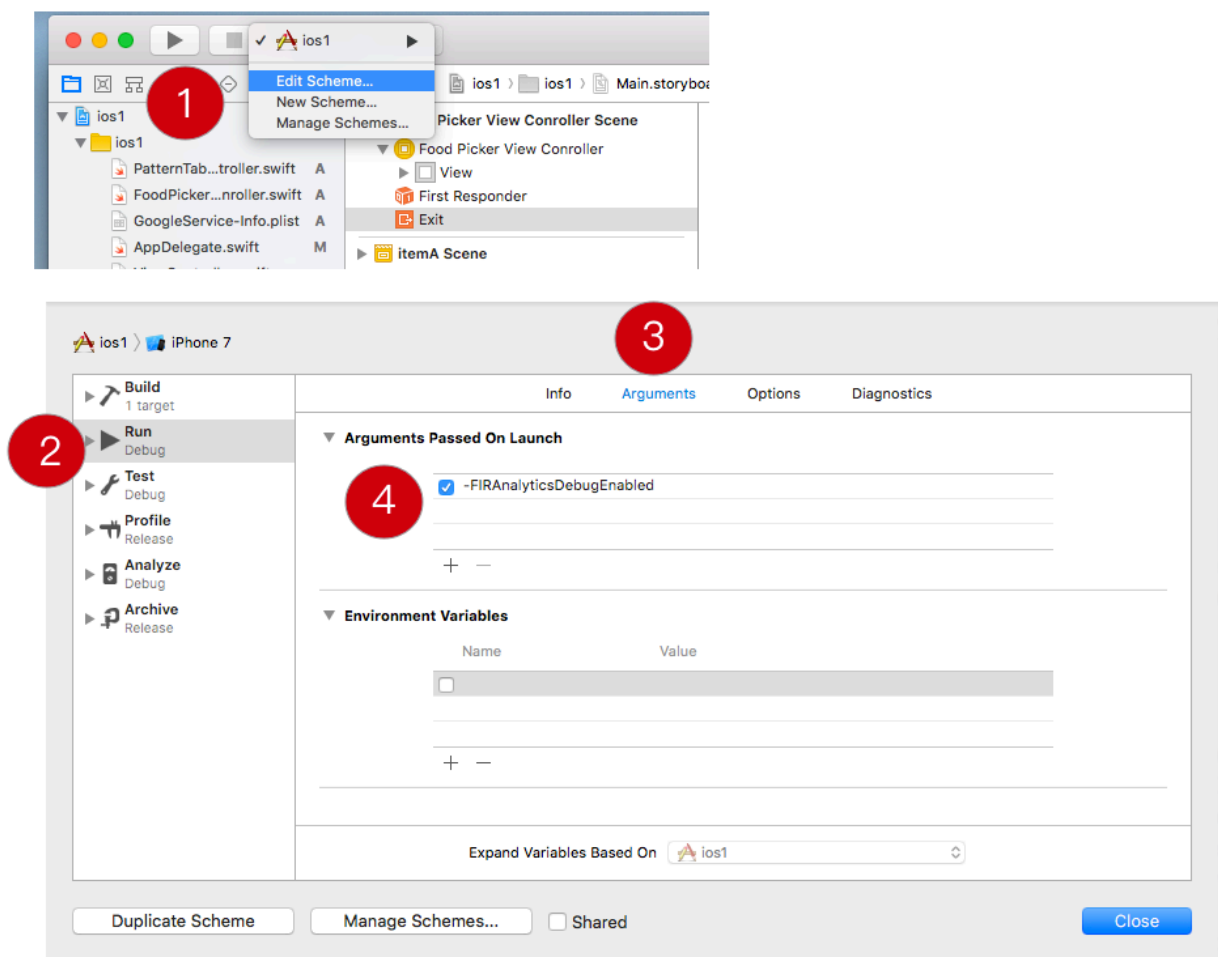
執行模擬器,選擇pizza。

在xcode debug


```
2017-10-04 14:55:32.327866+0800 ios1[3045:639597] [Firebase/Analytics][I-ACS023080] Setting user
property. Name, value: favorite_food, Pizza
2017-10-04 14:55:32.328473+0800 ios1[3045:639597] [Firebase/Analytics][I-ACS023087] User property set.
Name, value: favorite_food, Pizza
```

All Output ⌄   favorite_food   2
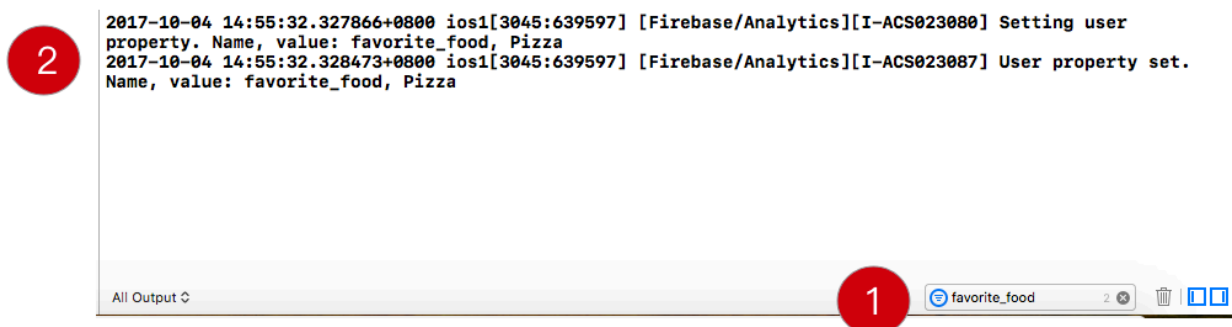
在Firebase consoled debugView及時觀察:
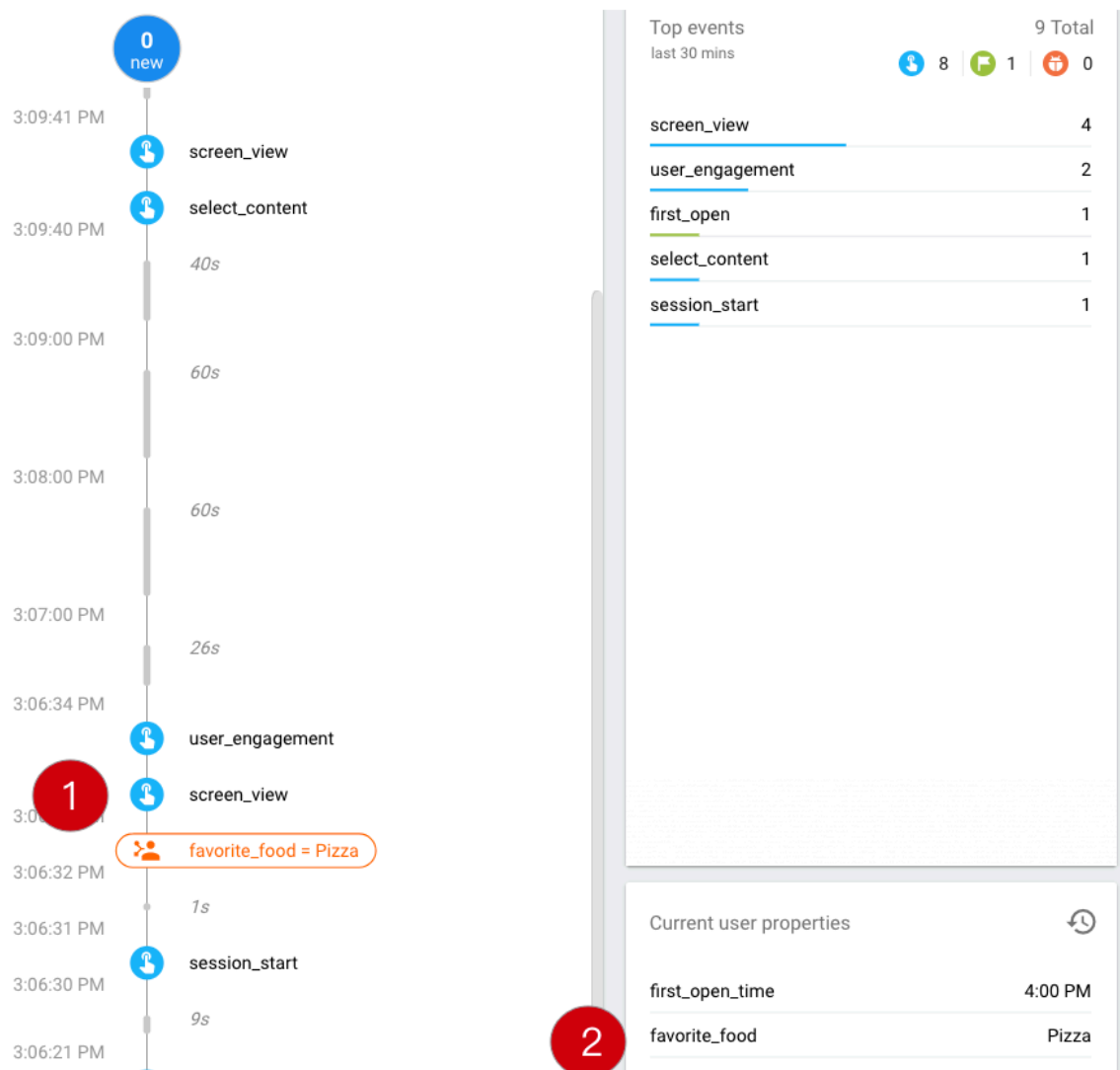
在Firebase console Events的filter內觀察(非及時，1日後觀察)

傳送Log Events

在storyboard設定4頁ViewController的title, 分別為itemA,itemB,itemC,itemD
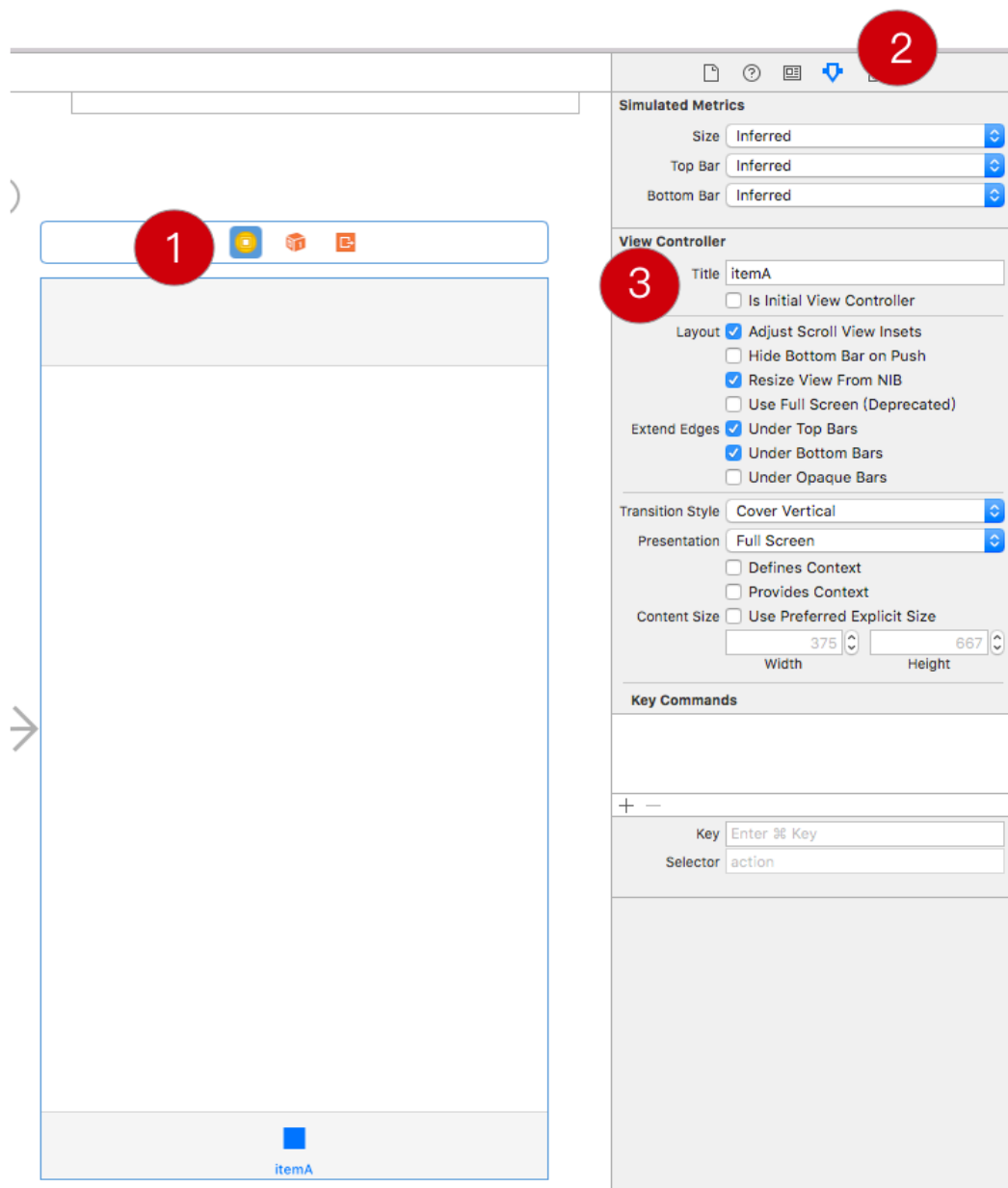
在每一頁顯示時，讓Firebase記錄這一頁的title和此頁的class Name

打開ViewController

```swift
1
2  import UIKit
3  import Firebase
4
5  class ViewController: UIViewController {
6
7
8      override func viewDidAppear(_ animated: Bool) {
9          super.viewDidAppear(animated);
10         recordScreenView();
11     }
12
13     override func didReceiveMemoryWarning() {
14         super.didReceiveMemoryWarning()
15         // Dispose of any resources that can be recreated.
16     }
17
18     func recordScreenView(){
19         guard let screenName = title else{
20             return;
21         }
22
23         let screenClass = classForCoder.description();
24
25         print("screenName:\(screenName)");
26         print("screenClass:\(screenClass)");
27
28         Analytics.setScreenName(screenName, screenClass: screenClass);
29     }
30
31
32  }
33
34
```

1

3 — 在viewDidAppear時間點，執行 recordScreenView()

取得頁面title

2

送出screen Name

在xcode-debug:

```
2017-10-04 15:55:59.347918+0800 ios1[4029:977742] [Firebase/Analytics][I-ACS023105] Event is not subject to real-time event count daily
limit. Marking an event as real-time. Event name, parameters: screen_view (_vs), {
    firebase_screen_id (_si) = -7410472433104522068;
    firebase_screen_class (_sc) = ios1.ViewController;
    firebase_screen (_sn) = itemA;
    firebase_realtime (_r) = 1;
    firebase_debug (_dbg) = 1;
    firebase_event_origin (_o) = auto;
}
```

在firebase-console -debugView

0
new

8s

3:56:20 PM

20s

3:56:00 PM

1s

3:55:59 PM

screen_view

3:55:58 PM

58s

3:55:00 PM

screen_view   ✕

Parameters   User properties

firebase_event_origin   auto

firebase_screen   itemA

firebase_screen_cla...   ios1.ViewController

firebase_screen_id   -74104724331045220...
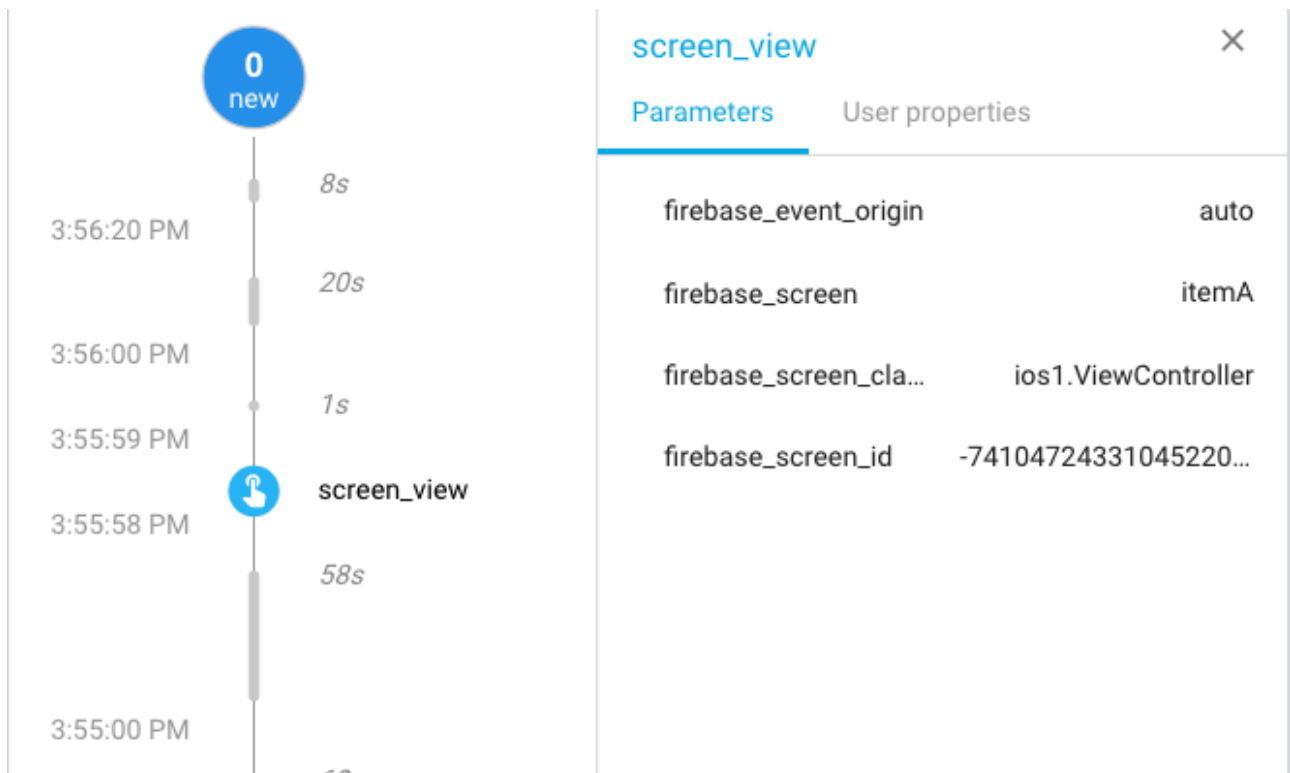
在firebase console dashboard內的顯示

User engagement > Screen class ⃝?

Daily engagement
3m 49s
-

Engagement per user
3m 49s
-

Sessions per user
2
-

Screen class	% Total engagement		Avg. engagement	
ios1.RestaurantTableViewCont...	50.4%	+83.6%	1m 10s	-69.1%
ios1.ViewController	24.5%	+10,030.9%	0m 30s	+26.4%
ViewController	9.3%		0m 9s	-

User engagement > Screen name ⃝?

Daily engagement
3m 49s
-

Engagement per user
3m 49s
-

Sessions per user
2
-

Screen name	% Total engagement		Avg. engagement	
(not set)	90.7%	-9.3%	0m 32s	-63.5%
A	6.1%	-	0m 19s	-
B	2.4%	-	0m 7s	-

送出內建的LogEvent，但夾帶自訂的參數

```swift
class ViewController: UIViewController {


    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated);
        recordScreenView();

        //自訂的 custom_event
        Analytics.logEvent(AnalyticsEventSelectContent, parameters: [
            AnalyticsParameterItemID: "id-\(title!)" as NSObject,
            AnalyticsParameterItemName:title! as NSObject,
            AnalyticsParameterContentType:"cont" as NSObject
            ]);
    }
}
```
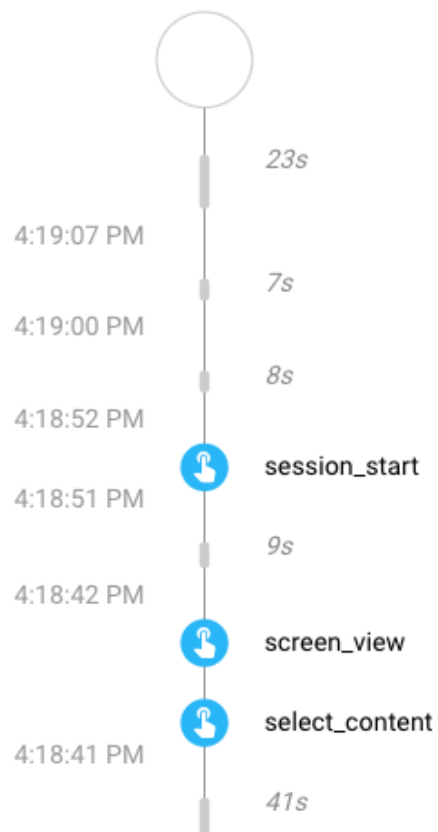
Xcode debug

```
2017-10-04 16:18:42.225079+0800 ios1[4173:1064579] [Firebase/Analytics][I-ACS023105] Eve
limit. Marking an event as real-time. Event name, parameters: select_content, {
    item_id = id-itemA;
    firebase_event_origin (_o) = app;
    item_name = itemA;
    firebase_realtime (_r) = 1;
    firebase_debug (_dbg) = 1;
    content_type = cont;
}
```
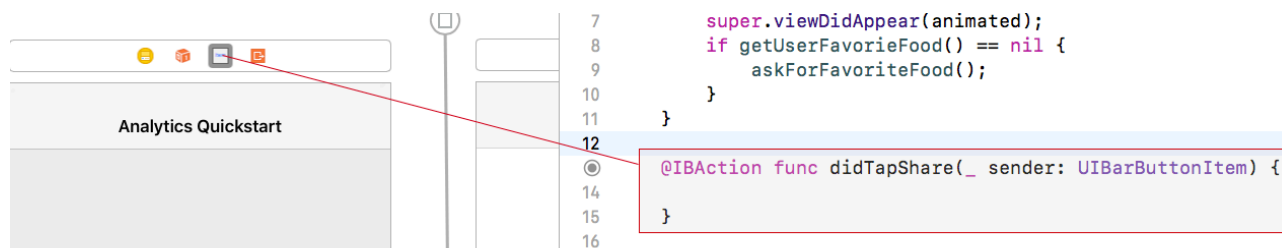
Firebase console debug view:

23s

4:19:07 PM

7s

4:19:00 PM

8s

4:18:52 PM

session_start

4:18:51 PM

9s

4:18:42 PM

screen_view

select_content

4:18:41 PM

41s

select_content	from 3:49 PM - 4:19 PM ✕

Parameters	User properties

content_type

content_type	cont
firebase_event_or...	app
item_id	id-itemA
item_name	itemA

▶ cont   4:18:41 PM

配合按鈕事件傳送LogEvent
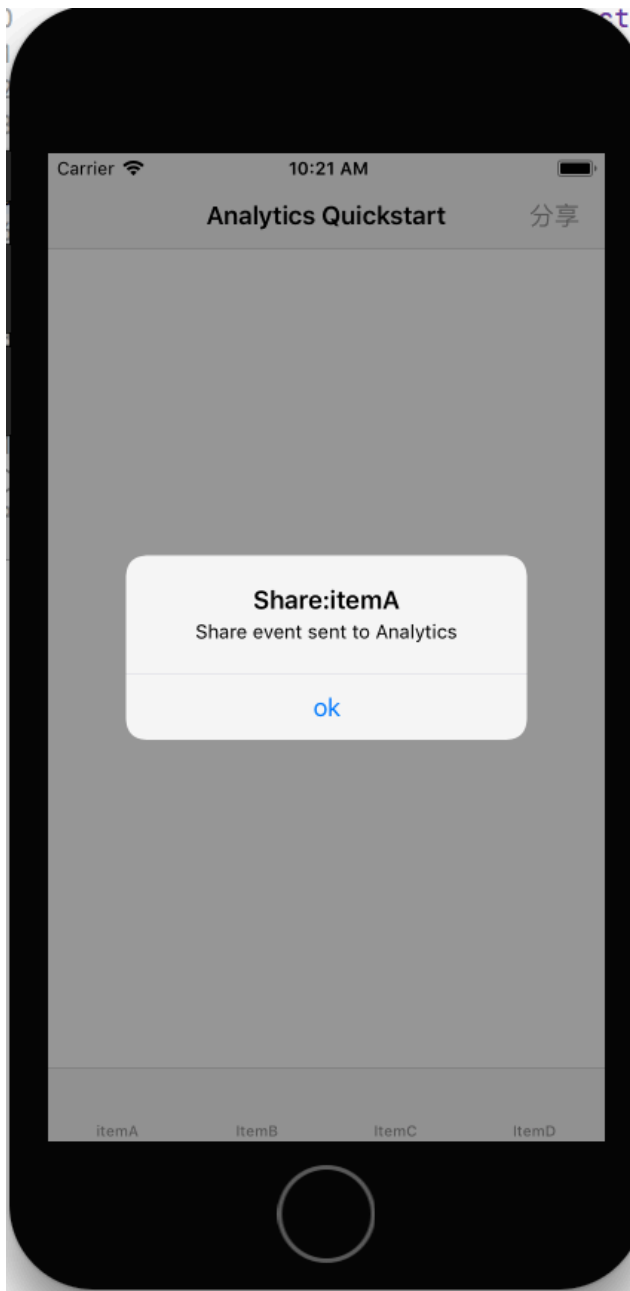打開 PatternTabBarController，建立分享按鈕的目標方法。

建立自訂的LogEvent

```swift
@IBAction func didTapShare(_ sender: UIBarButtonItem) {
    let name = "Pattern!\(self.selectedViewController!.title!)";
    let text = "I'd love you to hear about \(name)";

    //傳送出自訂的LogEvent
    Analytics.logEvent("share_image", parameters: [
        "name": name as NSObject,
        "full_text": text as NSObject
        ]);

    let title = "Share:\(self.selectedViewController!.title!)";
    let message = "Share event sent to Analytics";
    let aleartConroller = UIAlertController(title: title, message: message,
        preferredStyle: .alert);
    let alertAction = UIAlertAction(title: "ok", style: .default, handler: nil);
    aleartConroller.addAction(alertAction);
    present(aleartConroller, animated: true, completion: nil);
}
```
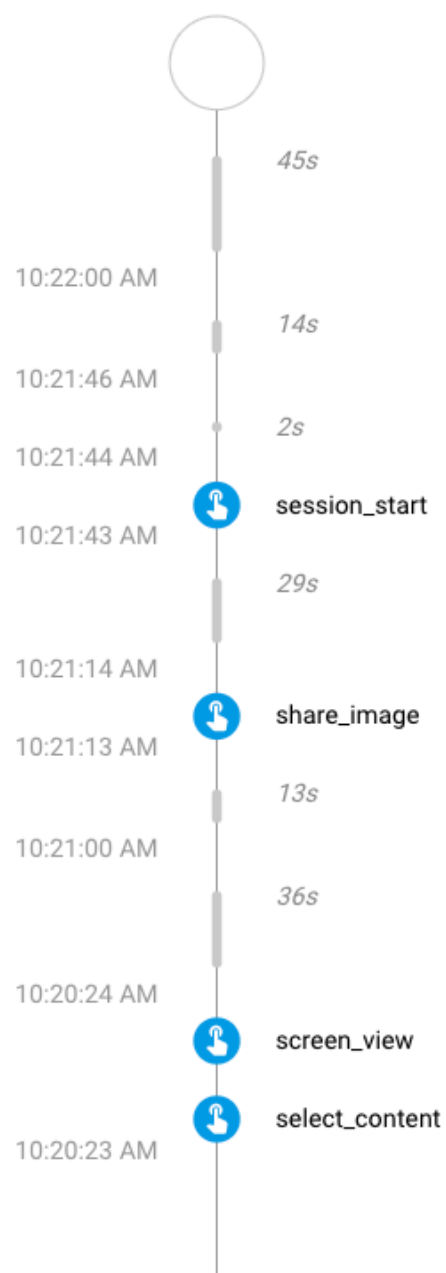
Alert對話框

執行並按下分享按鈕
模擬器顯示

Firebase console 的 debug view

45s

10:22:00 AM

14s

10:21:46 AM

2s

10:21:44 AM

session_start

10:21:43 AM

29s

10:21:14 AM

share_image

10:21:13 AM

13s

10:21:00 AM

36s

10:20:24 AM

screen_view

select_content

10:20:23 AM

Firebase console 的 debug view

share_image   from 9:52 AM - 10:22 AM   ✕

Parameters	User properties		firebase_event_origin	
firebase_event_origin	app		▶ app	10:21:13 AM
firebase_screen	itemA			
firebase_screen_cla...	ios1.ViewController			
firebase_screen_id	-34982011550097212...			
full_text	I'd love you to hear about Pattern!itemA			
name	Pattern!itemA			