POLITÉCNICA

ETSIT UPM

dit UPM

Desarrollo de Apps para iOS

Search Display Controller

IWEB-LSWC 2013-2014
Santiago Pavón

ver: 2013.10.19

Caso de Estudio: Películas

- Partiremos de una aplicación que muestra una tabla con un listado de películas.
  - Al seleccionar una película muestra los datos de la película en una nueva pantalla.
    - En realidad solo se muestra el título
- Modificaremos la aplicación para añadir una barra de búsqueda que nos permita localizar rápidamente una película en la tabla.


Carrier
6:46 PM
Películas
Lluvia de albóndigas
Infantil
Buscando a Nemo
Infantil
Bambi
Infantil
Indiana Jones y el templo mald…
Familiar
Mentiras arriesgadas
Familiar
La Rosa Púrpura del Cairo
Familiar
El juego de Ender
Familiar
Parque Jurásico
Familiar
El silencio de los corderos
Adultos
El exorcista
Carrier
6:49 PM
Películas
Parque Jurásico

Los Detalles de la App Original

La clase `Film`

- Las películas son objetos de la clase **`Film`**.
  - Tiene dos propiedades:
    - **`title`** - el título
    - **`genre`** - el género
      - **`Infantil`**, **`Familiar`**, **`Adultos`**.
  - Y un inicializador:
    - **`-initWithTitle:genre:`**

```
#import <Foundation/Foundation.h>

@interface Film : NSObject

@property (nonatomic, strong) NSString *title;
@property (nonatomic, strong) NSString *genre;

- (instancetype) initWithTitle:(NSString*)title
                         genre:(NSString*)genre;

@end
```

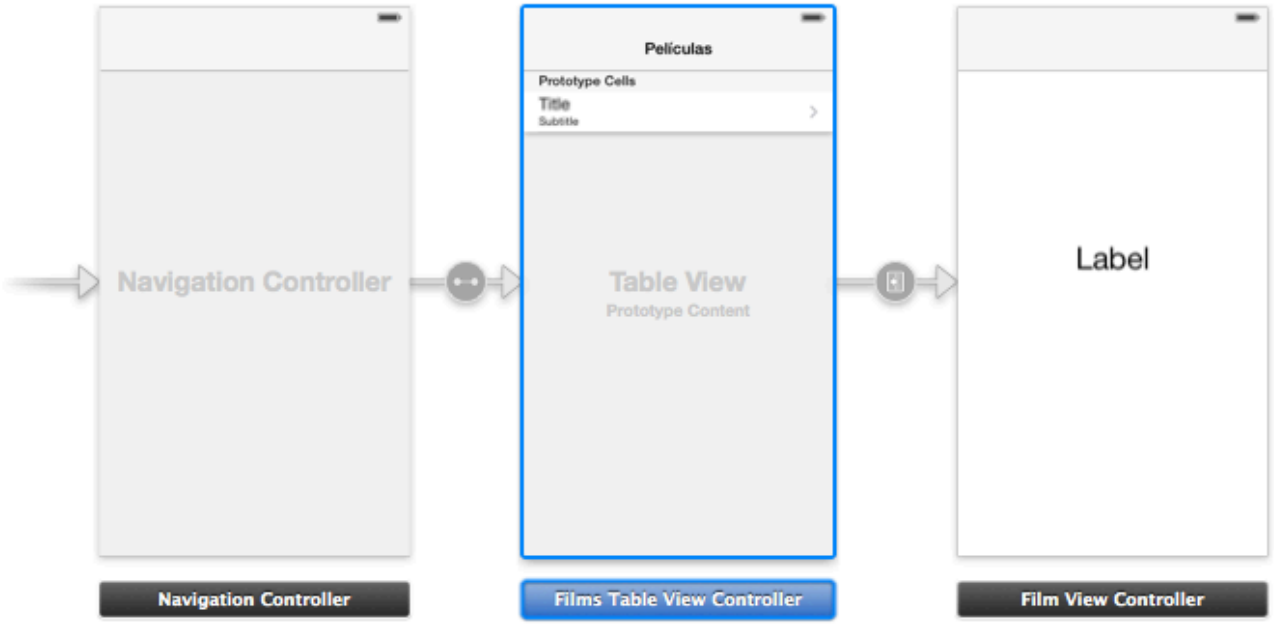
Film.h

```
#import "Film.h"

@implementation Film

- (instancetype) initWithTitle:(NSString*)title
                         genre:(NSString*)genre
{
    if (self = [super init]) {
        _title = title;
        _genre = genre;
    }
    return self;
}

@end
```

Film.m

Storyboard

- La aplicación usa un **`NavigationController`** con dos pantallas:
  - **`FilmsTableViewController`** para el listado de películas.
  - **`FilmViewController`** para ver los detalles de una película.



`FilmsTableViewController`

- Muestra el listado de películas.
- Detalles de la implementación:
  - Las películas se guardan en un `NSArray` que se crea en `viewDidLoad`.
  - Los métodos del protocolo **data source** simplemente acceden al array de películas.
    - Para ver el tamaño del array (= número de películas)
    - y construir las celdas de la tabla.
  - Solo hay un segue, **Show Film**, que pasa la película seleccionada a la segunda pantalla de detalles.

```
#import "FilmsTableViewController.h"
#import "FilmViewController.h"
#import "Film.h"


@interface FilmsTableViewController ()

@property (strong,nonatomic) NSArray *filmsArray; // Todas las peliculas

@end
```

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    // Todas las peliculas:
    self.filmsArray = @[[[Film alloc] initWithTitle:@"Lluvia de albóndigas"
                                              genre:@"Infantil"],
                        [[Film alloc] initWithTitle:@"Buscando a Nemo"
                                              genre:@"Infantil"],
                        [[Film alloc] initWithTitle:@"Bambi"
                                              genre:@"Infantil"],
                        [[Film alloc] initWithTitle:@"Indiana Jones y el templo maldito"
                                              genre:@"Familiar"],
                        [[Film alloc] initWithTitle:@"Mentiras arriesgadas"
                                              genre:@"Familiar"],
                        [[Film alloc] initWithTitle:@"La Rosa Púrpura del Cairo"
                                              genre:@"Familiar"],
                        [[Film alloc] initWithTitle:@"El juego de Ender"
                                              genre:@"Familiar"],
                        [[Film alloc] initWithTitle:@"Parque Jurásico"
                                              genre:@"Familiar"],
                        [[Film alloc] initWithTitle:@"El silencio de los corderos"
                                              genre:@"Adultos"],
                        [[Film alloc] initWithTitle:@"El exorcista"
                                              genre:@"Adultos"],
                        [[Film alloc] initWithTitle:@"El último tango en París"
                                              genre:@"Adultos"],
                        [[Film alloc] initWithTitle:@"Tiburón"
                                              genre:@"Adultos"]
                        ];
}
```

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger) tableView:(UITableView *)tableView
  numberOfRowsInSection:(NSInteger)section
{
    return [self.filmsArray count];
}

- (UITableViewCell *) tableView:(UITableView *)tableView
         cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellId = @"Film Cell";
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellId
                                                            forIndexPath:indexPath];

    Film *film = self.filmsArray[indexPath.row];

    cell.textLabel.text = film.title;
    cell.detailTextLabel.text = film.genre;

    return cell;
}
```

```
- (void) prepareForSegue:(UIStoryboardSegue *)segue
                  sender:(id)sender
{
    if ([segue.identifier isEqualToString:@"Show Film"]) {

        NSIndexPath *ip = [self.tableView indexPathForCell:sender];

        FilmViewController *fvc = segue.destinationViewController;

        fvc.film = self.filmsArray[ip.row];
    }
}
```

`FilmViewController`

- Muestra los detalles de una películas.
  - En realidad solo muestra su título.
- Detalles de implementación:
  - Hay una propiedad en el interface **`.h`** que actua como parámetro de entrada para obtener la película a mostrar.
  - La vista solo tiene una label para mostrar el título de la película.

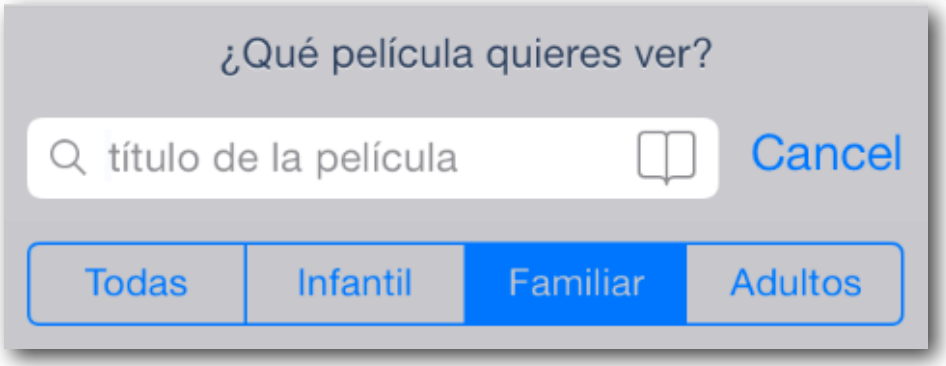
```
@interface FilmViewController : UIViewController

@property (nonatomic,strong) Film* film;

@end
```

```
@interface FilmViewController ()

@property (weak, nonatomic) IBOutlet UILabel *titleLabel;

@end

@implementation FilmViewController

- (void)viewDidLoad
{
    [super viewDidLoad];

    self.titleLabel.text = self.film.title;
}

@end
```

Añadir la Barra de Búsquedas

Barra de Búsqueda - `UISearchBar`

- Buscar un dato determinado en una tabla con muchas filas puede ser lento y frustrante.
- Las TableViews pueden incorporar una barra de búsqueda (**`UISearchBar`**) que permita filtrar la información mostrada en la tabla.
  - La información mostrada en la tabla se filtra continuamente adaptándose a lo que el usuario escribe en la barra de búsqueda.
- La barras de búsqueda también pueden opcionalmente mostrar una barra de ámbito (**scope bar**) para indicar el ámbito (tipo/categoría/clase) de los datos que se quieren buscar, un botón para cancelar, un botón para bookmarks, un prompt.



- Los objetos **`UISearchBar`** no realizan las búsquedas.
  - usan un delegado (**`UISearchBarDelegate`**) al que informan de lo que el usuario está escribiendo/pulsando, y delegan en él todo el trabajo.

Opciones de `UISearchBar`

- **`UISearchBar`** tiene muchas propiedades que se pueden configurar programáticamente o con Interface Builder.
  - **`text`**: texto a buscar.
  - **`placeholder`**: texto en gris que sirve de pista para el usuario.
  - **`prompt`**: prompt mostrado encima de la barra de búsqueda.
  - **`barStyle`**, **`searchBarStyle`**, **`barTintColor`**: apariencia.
  - **`showsSearchResultsButton`**: botón ver búsquedas o resultados recientes.
  - **`showsBookmarkButton`**: icono para mostrar los bookmarks salvados.
  - **`showsCancelButton`**: botón para cancelar.
  - **`showsScopeBar`**, **`scopeButtonTitles`**, **`selectedScopeButtonIndex`**: mostrar la barra de scopes, títulos de los botones de esta barra, e índice del botón seleccionado.
  - **`autocapitalizationType`**, **`autocorrectionType`**, **`keyboardType`**, **`spellCheckingType`**: configuración del campo de entrada de texto.
  - . . .

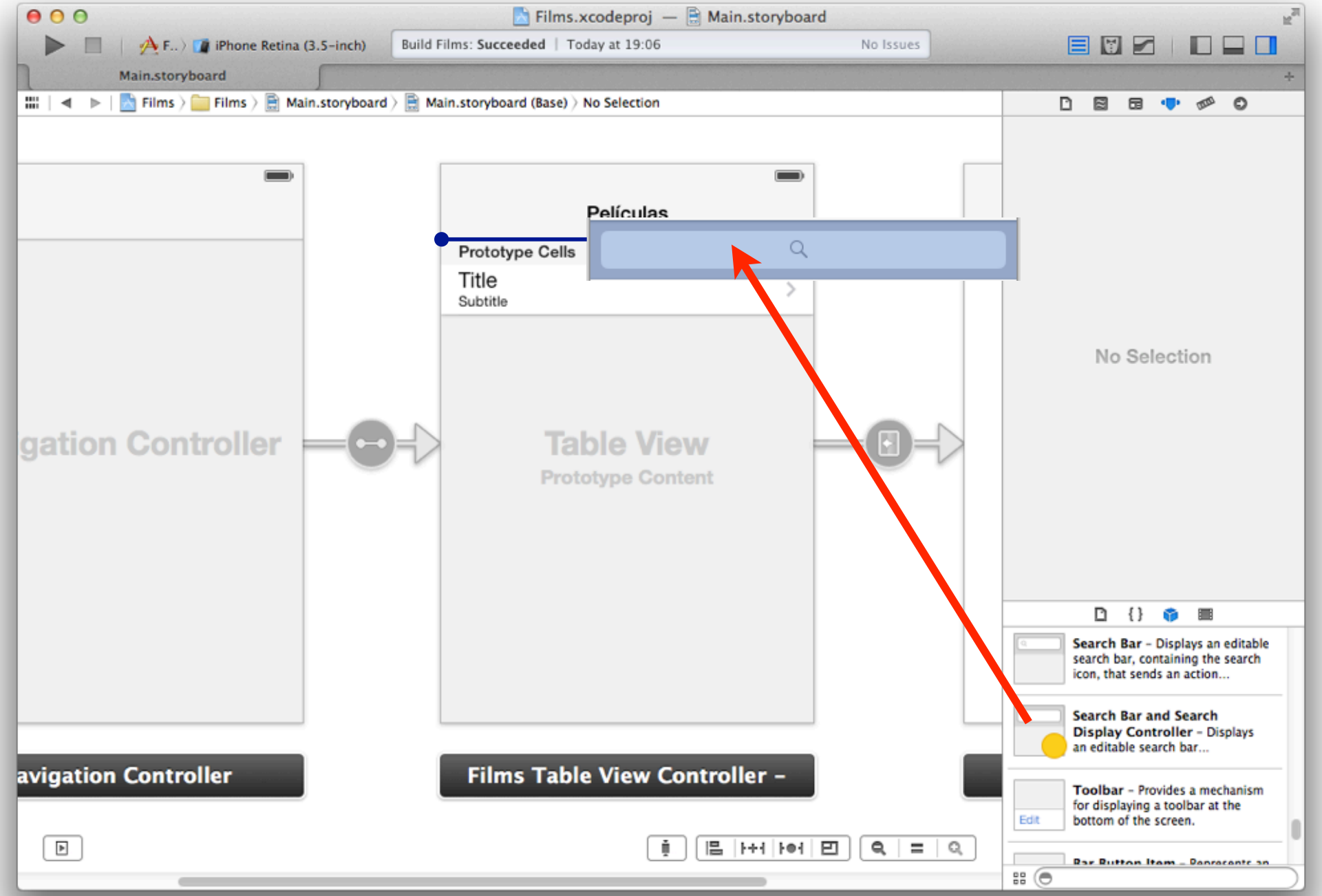
UISearchDisplayController

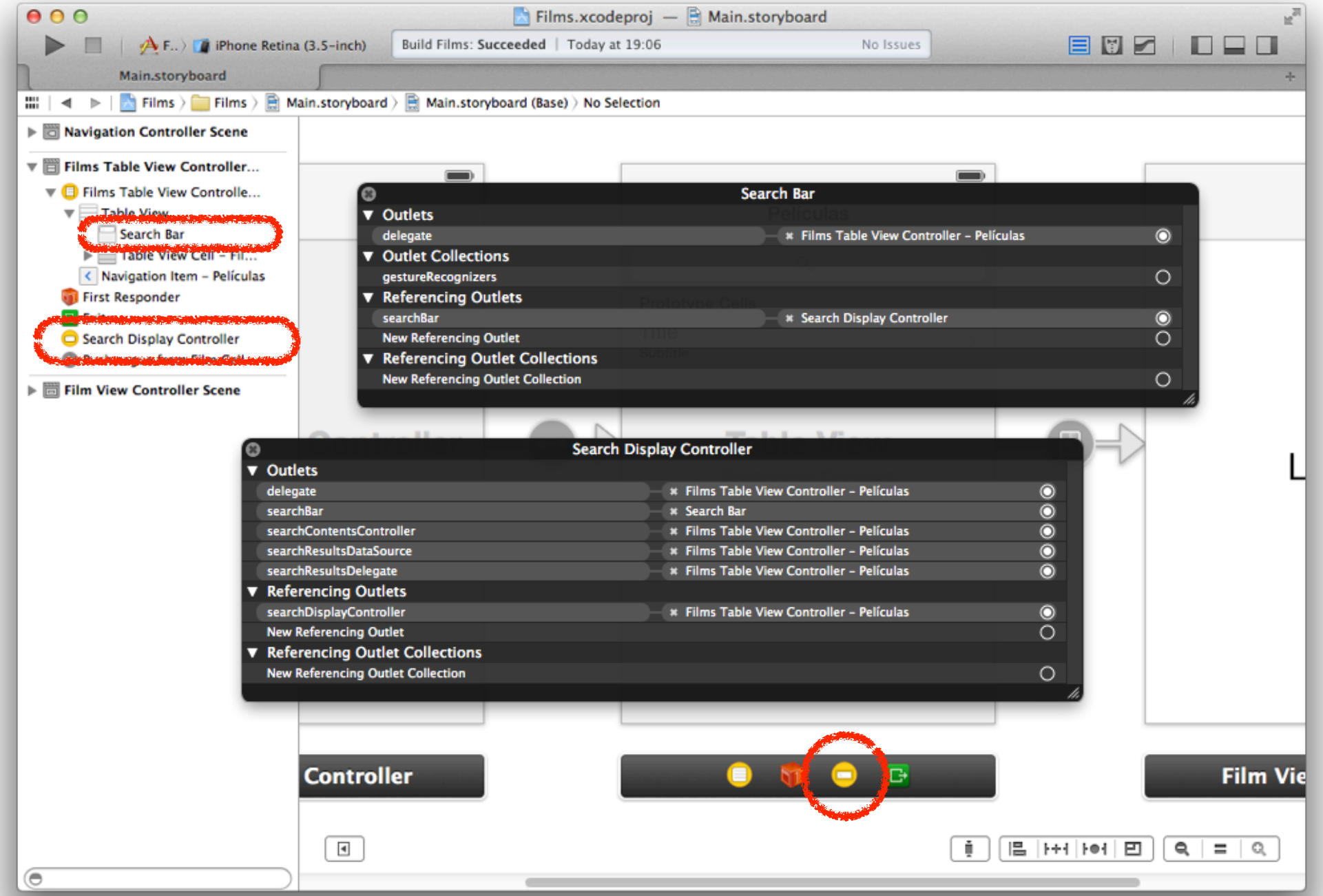
- Se inicializa con una **`UISearchBar`** y con el **`UIViewController`** responsable de los datos sobre los que se realizarán las búsquedas.
  - El SearchDisplayController crea internamente una TableView para mostrar el resultado de las búsquedas.
- Cuando el usuario inicia una búsqueda, el SearchDisplayController se encarga de superponer la SearchBar y la TablaView con el resultado de las búsquedas sobre el VC.
  - Nuestro código solo tiene que preocuparse de proporcionar los datos correctos (filtrados o sin filtrar) dependiendo de la tabla que se esté visualizando.
- El VC debe ampliarse para que realice **cuatro tareas**:
  - Debe ser el data source de la tabla creada por el SearchDisplayController.
  - Debe ser el delegado de la tabla creada por el SearchDisplayController.
  - Debe ser el delegado del propio SearchDisplayController.
  - Debe ser el delegado de la SearchBar.

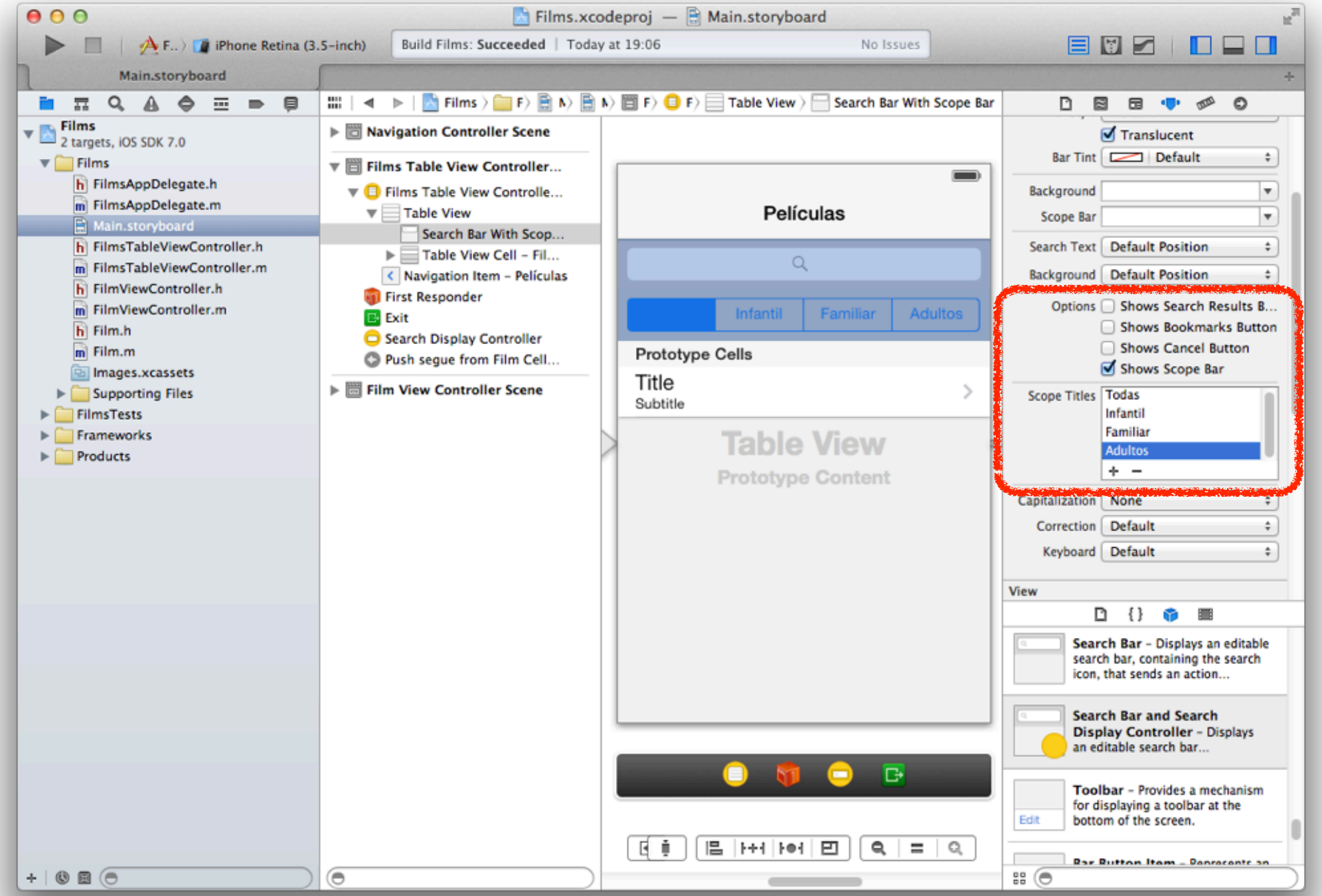
- **Propiedades**:
  - **`searchBar`** - la SearchBar.
  - **`searchContentsController`** - el VC propietario de los datos manejados.
  - **`searchResultsTableView`** - la TableView donde se muestra el resultado de las búsquedas.
  - **`searchResultsDataSource`** - el data source de la TableView de resultados.
  - **`searchResultsDelegate`** - el delegado de la TableView de resultados.
  - **`delegate`** - el delegado del propio SearchDisplayController.
  - **`searchResultsTitle`** - el título para tabla con los resultados.
  - **`displaysSearchBarInNavigationBar`** - booleano para indicar que la SearchBar se mostrará en la barra de navegación.
    - Las SearchBars presentadas en NavigationBars no pueden tener ScopeBar.
  - **`navigationItem`** - NavigationItem usado por el SearchDisplayController.
  - **`active`** - el estado de la interface de búsqueda.
- **Métodos**:
  - **`–initWithSearchBar:contentsController:`** - inicializador.
  - **`–setActive:animated:`** - cambiar el valor de **`active`** con una animación.

Crear con Interface Builder

- Trabajemos en el caso de estudio:
- Editamos el proyecto para añadir una barra de búsqueda a una TableViewController:
  - *Nótese que barra de búsqueda puede añadirse a cualquier ViewController, no tiene que se una TableVC.*
  - Embeber la TableVC en una NavigationBar, caso de que no lo esté.
  - Arrastrar desde la librería un objeto de tipo **Search Bar and Search Display Controller** hasta la TableVC,
    - **Cuidado**: No confundir el objeto **Search Bar and Search Display Controller** con un objeto **Search Bar**.
    - Posicionarlo entre la barra de navegación y la TableView.
    - Los outlets de la SearchBar y del SearchDisplayController se enlazan automáticamente con el TableViewController.
    - Editar las propiedades de UISearchBar para mostrar el ScopeBar, y crear los scopes: **Todas**, **Infantil**, **Familiar** y **Adultos**.







Delegados

- Editar la clase **`FilmsTableViewController`** para indicar que también es conforme con los protocolos:

  `UISearchBarDelegate`

  `UISearchDisplayDelegate`

  `UITableViewDelegate`

  `UITableViewDataSource`

  - En este caso de estudio, el ViewController con el que estamos trabajando es un UITableViewController.
    - Por tanto, ya es conforme por herencia con los dos últimos protocolos.
      - Entonces, solo hay que declarar los dos primeros.

```
@interface FilmsTableViewController ()
                <UISearchBarDelegate, UISearchDisplayDelegate>
```

Servir a Dos TableViews

- El objeto FilmsTableVC está actuando como el delegado y el data source de dos tablas:
  - La TableView del propio TableVC.

    ```
    self.tableView
    ```

  - La TableView de resultados que crea el SearchDisplayController.

    ```
    self.searchDisplayController.searchResultsTableView
    ```

- Es necesario retocar los métodos del **`UITableViewDelegate`** y del **`UITableViewDataSource`** para identificar cuál de las dos TableViews les ha invocado, y devolver los datos de la tabla correcta.
  - Puede consultarse el parámetro **`tableView`** de estos métodos para ver si es igual a `self.searchDisplayController.searchResultsTableView` o es igual a `self.tableView`.

- En nuestra app ya tenemos un `NSArray` con todas las películas manejadas por `FilmsTableViewCcontroller`.
- Ahora creamos otro `NSArray` para guardar el resultado de las búsquedas realizadas,

```
@interface FilmsTableViewController ()
        <UISearchBarDelegate, UISearchDisplayDelegate>

// Todas las peliculas:
@property (strong,nonatomic) NSArray *filmsArray;

// Resultados de las busquedas:
@property (strong,nonatomic) NSArray *filteredFilmsArray;

@end
```

  - Ahora:
    - Los métodos data source deben consultar el array **`filteredFilmsArray`** cuando se llamen para rellenar la tabla con los resultados de la búsqueda.
    - y deben consultar el array **`filmsArray`** cuando se estén obteniendo datos para `self.tableView`.

- Añadimos en **`viewDidLoad`**:

```
// Resultados de las busquedas:
self.filteredFilmsArray = [NSArray array];


// Registramos como se crean las celdas de la
// tabla de resultados.
[self.searchDisplayController.searchResultsTableView
                                registerClass:[UITableViewCell class]
                       forCellReuseIdentifier:@"Film Cell"];
```

Estamos registrando cómo se construyen las celdas de la tabla de búsquedas.

Al final del tema veremos como crear celdas personalizadas para las dos tablas.

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return 1;
}

- (NSInteger) tableView:(UITableView *)tableView
  numberOfRowsInSection:(NSInteger)section {

    // Mirar que array debo consultar: el array normal o el de resultados de busquedas:
    if (tableView == self.searchDisplayController.searchResultsTableView) {
        return [self.filteredFilmsArray count];
    } else {
        return [self.filmsArray count];
    }
}

- (UITableViewCell *)tableView:(UITableView *)tableView
         cellForRowAtIndexPath:(NSIndexPath *)indexPath {

    static NSString *CellId = @"Film Cell";
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellId];

    Film *film;
    // Mirar que array debo consultar: el array normal o el de resultados de busquedas:
    if (tableView == self.searchDisplayController.searchResultsTableView) {
        film = self.filteredFilmsArray[indexPath.row];
    } else {
        film = self.filmsArray[indexPath.row];
    }

    cell.textLabel.text = film.title;
    cell.detailTextLabel.text = film.genre;

    return cell;
}
```

¿Cómo se filtra?

- Necesitamos un método auxiliar para calcular el valor de `self.filteredDataArray`.
  - Este método toma como parámetros los datos de lo que se desea buscar, y devuelve un array con las películas que cumplan con el criterio de búsqueda.
  - Este método puede implementarse usando NSPredicate, o recorriendo el array original analizando todas las posiciones.
  - Usaré `–filteredArrayUsingPredicate:` para obtener un array filtrado con las películas que nos interesan.

```
-(void) updateFilteredFilmsWithTitle:(NSString*)text
                               genre:(NSString*)genre
{
    // Asigno a self.filteredFilmsArray un array con las peliculas
    // que contengan "text" en su titulo y que sean del genero "genre".

    // Acepto genre igual a nil por si no uso scopes.

    // Busco usando un NSPredicate.
    // - "SELF.title" indica buscar en la propiedad "title".
    // - “contains[c]” indica contener sin distinguir mayusculas
    //                  de minusculas.
    // - "%@" se sustituye por el texto a buscar.
    NSPredicate *pred = [NSPredicate predicateWithFormat:
                                    @"SELF.title contains[c] %@",text];

    NSArray *tmp = [self.filmsArray filteredArrayUsingPredicate:pred];

    if (genre && ![genre isEqualToString:@"Todas"]) {
        NSPredicate *pred2 = [NSPredicate predicateWithFormat:
                                        @"SELF.genre contains[c] %@",genre];
        tmp = [tmp filteredArrayUsingPredicate:pred2];
    }

    self.filteredFilmsArray = tmp;
}
```

Actualizar la Búsqueda

- Cuando se cambia el texto a buscar, o se cambia el ámbito (scope) de búsqueda, se invocan estos métodos en el delegado del SearchDisplayController:
  - `- (BOOL) searchDisplayController:(UISearchDisplayController*)sdc shouldReloadTableForSearchString:(NSString*)searchString`
  - `- (BOOL) searchDisplayController:(UISearchDisplayController*)sdc shouldReloadTableForSearchScope:(NSInteger)searchOption`
- Estos métodos deben realizar una nueva búsqueda y devolver **`YES`** para recargar la tabla con los resultados de la búsqueda.
  - Para hacer una búsqueda asíncrona hay que devolver **`NO`**.
    - Cuando se obtengan los datos se forzará la recarga de la tabla de resultados.

```
// Ha cambido el texto a buscar.
-(BOOL)    searchDisplayController:(UISearchDisplayController*)controller
  shouldReloadTableForSearchString:(NSString *)searchString
{
    NSString * genre;
    if (self.searchDisplayController.searchBar.showsScopeBar) {
        NSUInteger index =
             self.searchDisplayController.searchBar.selectedScopeButtonIndex;
        genre =
             self.searchDisplayController.searchBar.scopeButtonTitles[index];
    }
    [self updateFilteredFilmsWithTitle:searchString
                                 genre:genre];
    return YES;
}


// Ha cambiado el scope a buscar.
-(BOOL)    searchDisplayController:(UISearchDisplayController*)controller
   shouldReloadTableForSearchScope:(NSInteger)searchOption
{
    NSString * genre =
      self.searchDisplayController.searchBar.scopeButtonTitles[searchOption];
    NSString * title = self.searchDisplayController.searchBar.text;

    [self updateFilteredFilmsWithTitle:title
                                 genre:genre];
    return YES;
}
```

Lanzar el Segue

- Ahora queremos saltar a la pantalla de detalles al tocar en cualquiera de las dos tablas:
  - la tabla de todas las películas.
  - la tabla con los resultados de las búsquedas.
- Cambiamos lo siguiente.
  - El segue lo cambiamos para que su origen sea **`FilmsTableVC`**, y no las celdas de **`self.tableView`**.
  - Sobreescribimos el método **`-tableView:didSelectRowAtIndexPath:`** para lanzar el segue programáticamente cuando se toque una celda de cualquiera de las dos tablas.
  - Retocamos **`-prepareForSegue:sender:`** para ver cuál es la tabla con la que estamos trabajando, y así sacar la película que hay que pasar a la siguiente pantalla de la tabla adecuada.

```
- (void)       tableView:(UITableView *)tableView
 didSelectRowAtIndexPath:(NSIndexPath *)indexPath {

    [self performSegueWithIdentifier:@"Show Film"
                              sender:tableView];
}


- (void) prepareForSegue:(UIStoryboardSegue *)segue
                  sender:(id)sender
{
    if ([segue.identifier isEqualToString:@"Show Film"]) {

        FilmViewController *fvc = segue.destinationViewController;

        if (sender == self.searchDisplayController.searchResultsTableView) {

            NSIndexPath *ip = [self.searchDisplayController.searchResultsTableView
                                                        indexPathForSelectedRow];

            fvc.film = self.filteredFilmsArray[ip.row];

        } else {

            NSIndexPath *ip = [self.tableView indexPathForSelectedRow];

            fvc.film = self.filmsArray[ip.row];
        }
    }
}
```

Uso en un NavigationBar

- Desde iOS7 se puede usar un SearchDisplayController con una NavigationBar.
- Hay que hacer los siguentes cambios:
  - Con Interface Builder: mover el objeto SearchBar fuera de la view.
  - Añadir en `viewDidLoad` la sentencia:

```
self.searchDisplayController.displaysSearchBarInNavigationBar = YES;
```

    - Con esto se coloca la SearchBar en la barra de navegación y se oculta el boton ***Cancel***.
- Nota: Cuando se coloca la SearchBar en la NavigationBar, no puede usarse la ScopeBar.
  - Desmarcar esta opción de la SearchBar con el inspector de atributos.

- Para que el botón ***Cancel*** se muestre y oculte automáticamente hay que añadir estos métodos del delegado de SearchDisplayController:

```
- (void)searchDisplayControllerWillBeginSearch:(UISearchDisplayController*)sdc
{
  self.searchDisplayController.searchBar.showsCancelButton = YES;
}

- (void)searchDisplayControllerDidEndSearch:(UISearchDisplayController*)sdc
{
  self.searchDisplayController.searchBar.showsCancelButton = NO;
}
```

Celdas Personalizadas

- Las celdas de una tabla se crean en el método **`tableView:cellForRowAtIndexPath:`** de su data source.
  - La tabla **`self.tableView`** crea celdas nuevas copiando el prototipo diseñado en el storyboard.
  - Pero las celdas de la tabla de las búsquedas no se crean usando el prototipo del storyboard.
- Las celdas de la tabla de búsquedas las tenemos que crear:
  - Programáticamente: cuando no haya celdas que reutilizar, creamos una celda nueva con esta sentencia.

```
cell = [UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
                                reuseIdentifier:@"Film Cell"];
```

    - También podría crearse la celda usando una clase derivada.
  - Registrando una clase o un fichero NIB para crear las celdas automáticamente cuando no haya celdas que reutilizar.
    - Para crear las celdas usando la clase **`UITableViewCell`** añadimos en **`viewDidLoad`**:

```
[self.searchDisplayController.searchResultsTableView
                                registerClass:[UITableViewCell class]
                        forCellReuseIdentifier:@"Film Cell"];
```

      - Las celdas se crean usando el estilo Básico de celda.
    - Para crear las celdas usando el diseño de un fichero NIB, añadimos en **`viewDidLoad`**:

```
UINib * nib = [UINib nibWithNibName:@"FilmCell" bundle:nil];
[self.searchDisplayController.searchResultsTableView
                                registerNib:nib
                        forCellReuseIdentifier:@"Film Cell"];
```

- Para que las celdas de las dos tablas sean iguales , podemos crear un diseño personalizado de celda en un fichero NIB, y registrar este fichero NIB para crear las celdas de las dos tablas.
  - Añadiríamos en **`viewDidLoad:`**

```
UINib * nib = [UINib nibWithNibName:@"FilmCell" bundle:nil];

[self.searchDisplayController.searchResultsTableView
                                        registerNib:nib
                              forCellReuseIdentifier:@"Film Cell"];

[self.tableView registerNib:nib
     forCellReuseIdentifier:@"Film Cell"];
```

- Los pasos para crear un fichero NIB para las celdas de una tabla son:
  - Crear un fichero NIB vacío desde:
    - **`Menu File > New > File... > iOS  User Interface > Empty`**
  - Con Interface Builder, añadir al NIB vacío un objeto `UITableViewCell` desde la librería de objetos.
  - Crear una nueva clase Objective-C derivada de `UITableViewCell` para la celda.
  - Diseñar el contenido de la celda y crear los outlets que se necesiten.