ETSIT UPM

dit UPM

Desarrollo de Apps para iOS

Split View Controller y Popover Controller

IWEB,LSWC 2013-2014

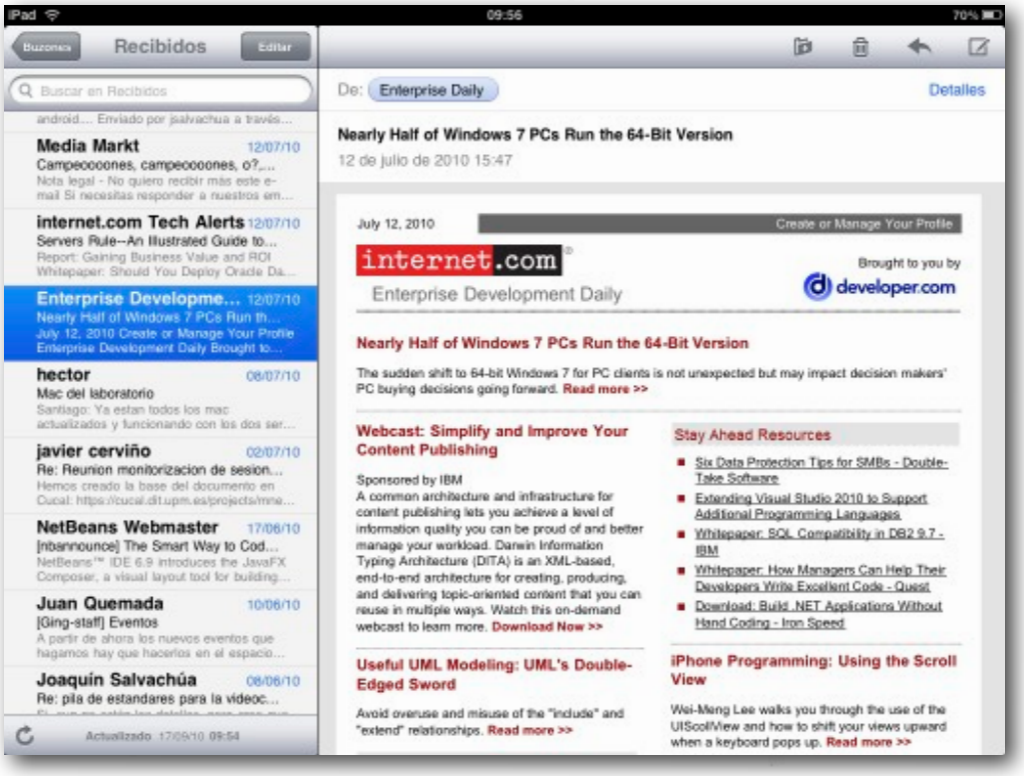
Santiago Pavón

ver: 2013.10.19

Split View Controller

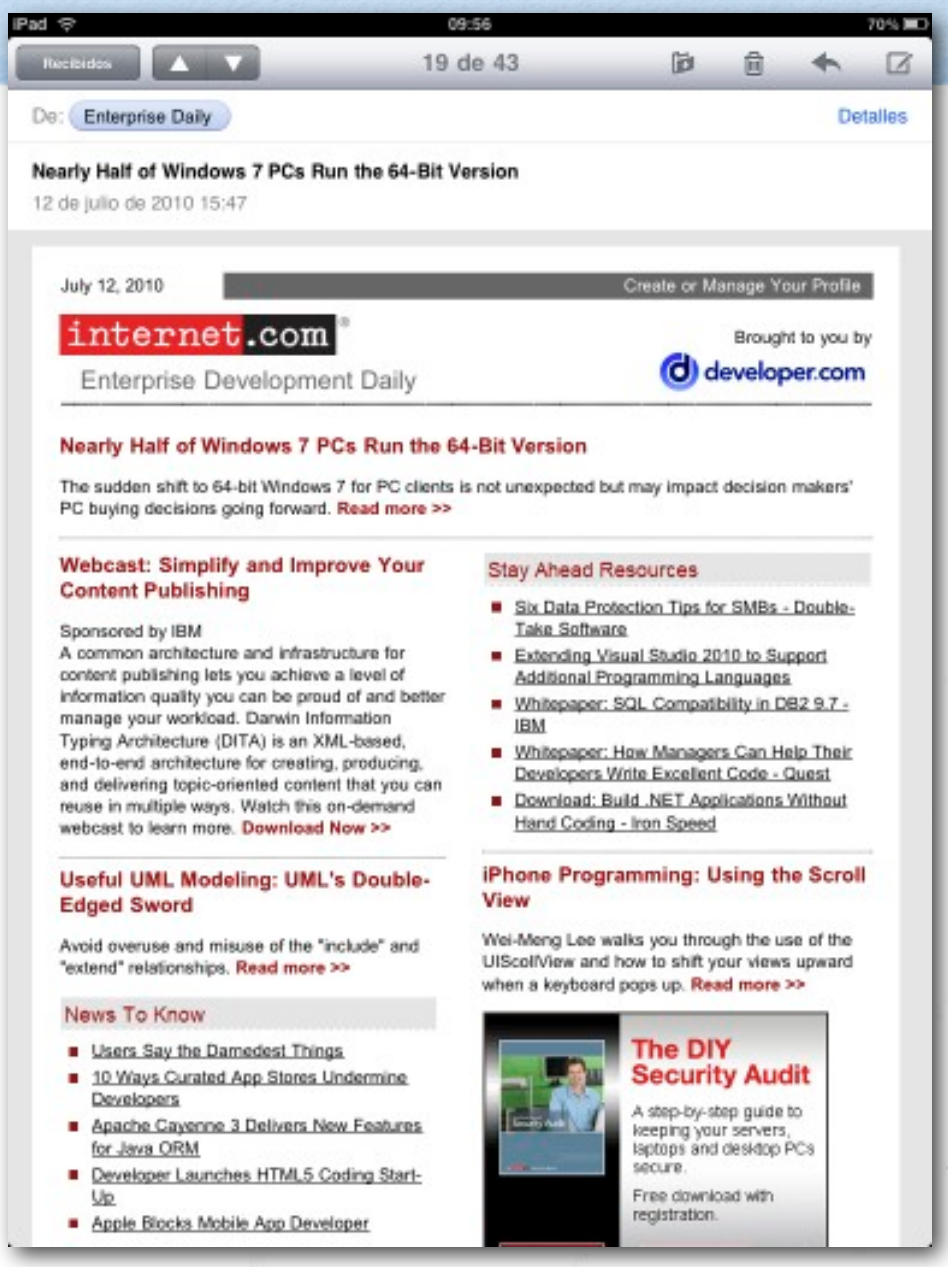
UISplitViewController

- En posición horizontal (landscape):
  - Muestra dos View Controllers:
    - en la izquierda: el VC master *(podría no mostrarse)*.
    - en la derecha: el VC detail.



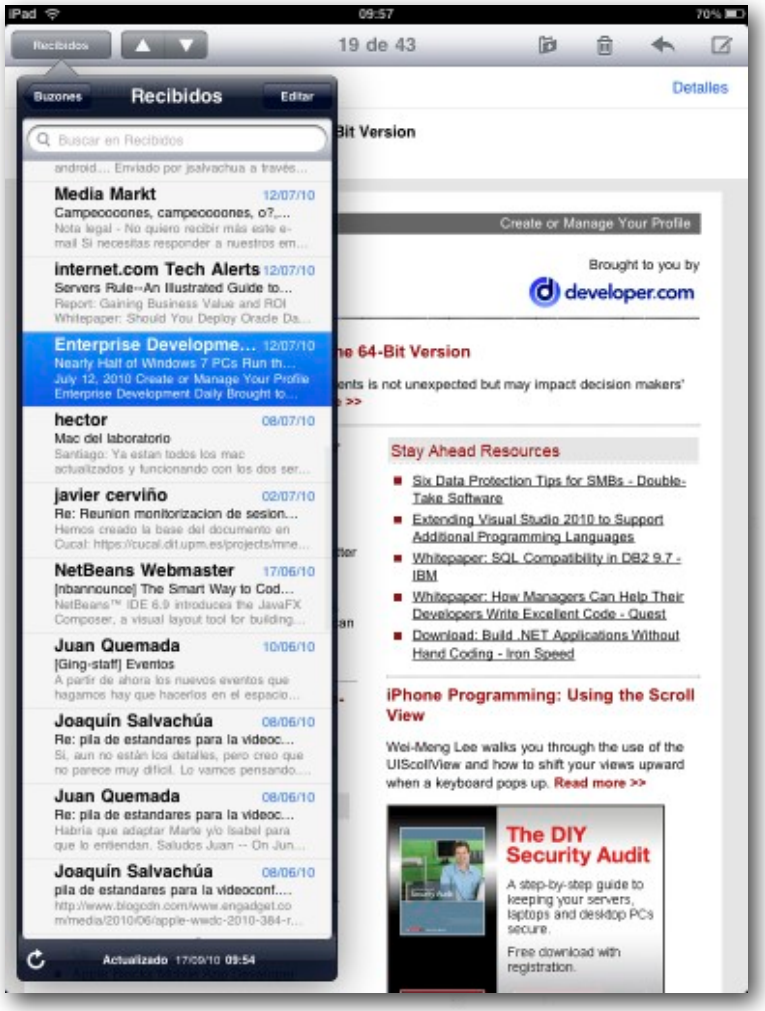
UISplitViewController

- En posición vertical:
  - Muestra el VC detail.



UISplitViewController

- En posición vertical:
  - Muestra VC detail.
  - El VC master puede desplegarse en un popover.



Crear un `UISplitViewController`

- **Programáticamente**:

```
[[UISplitViewController alloc] init];
```

- Editando un **Storyboard** que haya sido creado para iPad:
  - Hay que arrastrar un objeto Split View Controller desde la librería de objetos.
  - Si el storyboard fue creado para iPhone, entonces en la librería de objetos no aparecen los objetos de tipo Split View Controller.
- Xcode proporciona una **plantilla** para crear nuevos proyectos basados en Split View Controller.
- Normalmente, en las aplicaciones que usan Split View Controller:
  - Sólo existe un único Split View Controller en la aplicación.
  - Es el View Controller inicial de la aplicación.
    - No estará embebido dentro de otro View Controller.
  - Sus VC master y detail seguramente usarán internamente algún otro controlador.
    - por ejemplo un Navigation Controller.

Propiedades de UISplitViewController

```
@property (nonatomic,copy) NSArray *viewControllers
```

- Este array tiene dos elementos:
  - índice 0: el VC master
  - índice 1: el VC detail
- Nótese el uso de la opción **`copy`**: se devuelve o se asigna una copia.
  - No podemos modificar el array directamente.

```
@property (nonatomic,assign) id<UISplitViewControllerDelegate>
delegate
```

- Se usa para preguntar en que orientaciones debe ocultarse el VC master.
- También se usa para avisar cuando se va mostrar o ocultar el VC master
- Para gestionar el BarButton que muestra el VC master en un popover.
- Nótese el uso de la opción **`assign`**: no se asigna `nil` cuando el delegado se destruye.

```
@property (nonatomic) BOOL presentsWithGesture
```

- Indica si el VC master puede presentarse y ocultarse con un gesto swipe.
  - El delegado debe implementar el método **`splitViewController:willHideViewController:withBarButtonItem:forPopoverController:`** .

Propiedades de UIViewController

- En UISplitViewController.h se define una categoría que extiende la clase `UIViewController` con una nueva propiedad:

```
@property (nonatomic, readonly, retain)
        UISplitViewController *splitViewController
```

  - Esta propiedad apunta al objeto Split View Controller que contiene al VC.
    - Es `nil` si el VC no está contenido en un Split View Controller.

UISplitViewControllerDelegate

- Este delegado decide si se muestra o no el VC master; gestiona el popover usado para mostrar el VC master; y gestiona el botón que hay que usar para abrir el popover que muestra el VC master.

```
- (BOOL)splitViewController:(UISplitViewController *)svc
   shouldHideViewController:(UIViewController *)vc
              inOrientation:(UIInterfaceOrientation)orientation

- (void)splitViewController:(UISplitViewController*)svc
          popoverController:(UIPopoverController*)pc
  willPresentViewController:(UIViewController *)aViewController

- (void)splitViewController:(UISplitViewController*)svc
     willHideViewController:(UIViewController *)aViewController
          withBarButtonItem:(UIBarButtonItem*)barButtonItem
       forPopoverController:(UIPopoverController*)pc

- (void)splitViewController:(UISplitViewController*)svc
     willShowViewController:(UIViewController *)aViewController
  invalidatingBarButtonItem:(UIBarButtonItem *)button
```

- En iOS 7 se añadieron nuevos métodos para que el delegado indique las orientaciones soportadas por el SplitVC y la orientación preferida:

```
- (NSUInteger)splitViewControllerSupportedInterfaceOrientations:
                        (UISplitViewController*)splitViewController
```

  - Devuelve un valor de los definidos en el enum **`UIInterfaceOrientationMask`**
    - `UIInterfaceOrientationMaskPortrait`
    - `UIInterfaceOrientationMaskLandscapeLeft`
    - `UIInterfaceOrientationMaskLandscapeRight`
    - `UIInterfaceOrientationMaskPortraitUpsideDown`
    - `UIInterfaceOrientationMaskLandscape`
    - `UIInterfaceOrientationMaskAll`
    - `UIInterfaceOrientationMaskAllButUpsideDown`

```
- (UIInterfaceOrientation)
  splitViewControllerPreferredInterfaceOrientationForPresentation:
                        (UISplitViewController *)splitViewController
```

  - Devuelve un valor de los definidos en el enum **`UIInterfaceOrientation`**
    - `UIInterfaceOrientationPortrait`
    - `UIInterfaceOrientationPortraitUpsideDown`
    - `UIInterfaceOrientationLandscapeLeft`
    - `UIInterfaceOrientationLandscapeRight`

Visibilidad del VC Master

- Mostrar o no el VC master dependiendo de la orientación es un comportamiento que puede cambiarse sobreescribiendo en el delegado el siguiente método:

```
-(BOOL)splitViewController:(UISplitViewController *)svc
  shouldHideViewController:(UIViewController *)vc
             inOrientation:(UIInterfaceOrientation)orientation;
```

  - Cuando no queremos mostrar el master VC, nos proporcionarán un botón que pondremos en una barra de herramientas, y que al ser pulsado mostrará el VC master en un popover.
  - Este método devolverá:
    - **NO**
      - Para que el masterVC se muestre con el terminal en posición vertical y apaisada.
    - **YES**
      - No se muestra en ninguna orientación.
    - **UIInterfaceOrientationIsPortrait(***el_argumento_orientation***)**
      - Solo se muestra con el terminal apaisado.
    - **...**

- Si al rotar el terminal se va a ocultar el VC master, se invoca antes el siguiente método del delegado:

```
- (void)splitViewController:(UISplitViewController *)svc
     willHideViewController:(UIViewController *)aViewController
          withBarButtonItem:(UIBarButtonItem *)barButtonItem
       forPopoverController:(UIPopoverController *)pc;
```

  - En el tercer parámetro de este método nos proporcionan un Bar Button que al ser pulsado mostrará un popover conteniendo el VC master.
    - Para poder usar este Bar Button, debemos colocarlo en algún lugar de la interface.
      - Típicamente se suele colocar en una UIToolBar o en una UINavigatioBar dentro del VC detail.
  - En el cuarto parámetro es el popover donde se mostrará el Master VC al tocar el Bar Button Item dado.

- Si al rotar el terminal se va a mostrar el VC master que estaba oculto, se invoca antes el siguiente método del delegado:

```
- (void)splitViewController:(UISplitViewController *)svc
     willShowViewController:(UIViewController *)aViewController
  invalidatingBarButtonItem:(UIBarButtonItem *)button;
```

  - El Bar Button que nos pasan como tercer parámetro es el que nos proporcionaron anteriormente para hacer que se mostrara el VC master.
    - Este Bar Button ya no es válido y debemos eliminarlo de la interface.

- Cuando pulsamos el Bar Button descrito anteriormente, el VC master se muestra dentro de un Popover Controller.

  - Antes de mostrarse se invoca el siguiente método del delegado:

```
- (void)splitViewController:(UISplitViewController *)svc
          popoverController:(UIPopoverController *)pc
  willPresentViewController:(UIViewController *)aViewController
```

  - Este método puede usarse para hacer alguna configuración en el VC Master antes de que aparezca.

Asignar el delegado

- La asignación del delegado del SplitVC debe hacerse antes de que el SplitVC empiece a enviar mensajes a su delegado.
  - Para **evitar que se pierdan mensajes** enviados a un delegado **`nil`**.
- ¿Donde podemos asignar el delegado?
  - La asignación del delegado del SplitVC puede hacerse en el método **`application:didFinishLaunchingWithOptions:`** del ApplicationDelegate.
    - Así lo hace la plantilla generada por Xcode.
  - También puede hacerse en el método **`awakeFromNib`** del objeto a usar como delegado.

```
- (void) awakeFromNib {
    self.splitViewController.delegate = self;
}
```

Actualizar el VC Details

- En el Master VC tendremos controles/gestos que actualicen el Detail VC.
- Formas de actualizar el Detail VC:
  1. El Master VC ejecuta algún método o cambia el valor de alguna propiedad del objeto Detail VC.
     - En estos casos no se destruye el objeto Detail VC existente, solo se le cambia algo interno.
       - Se sigue mostrando el mismo objeto Detail VC.
  2. El Master VC dispara un segue de tipo **replace**.
     - En este caso se crea un nuevo objeto VC details.
       - El antiguo Detail VC se destruye.
     - Hay que recolocar en el nuevo Detail VC el Bar Button que nos proporcionaron para abrir el popover que muestra el Master VC.
     - Pensar ¿Quien era el delegado del SplitVC? ¿Era el Detail VC destruido?

Colocación de un BarButtonItem

- En un NavigationBar

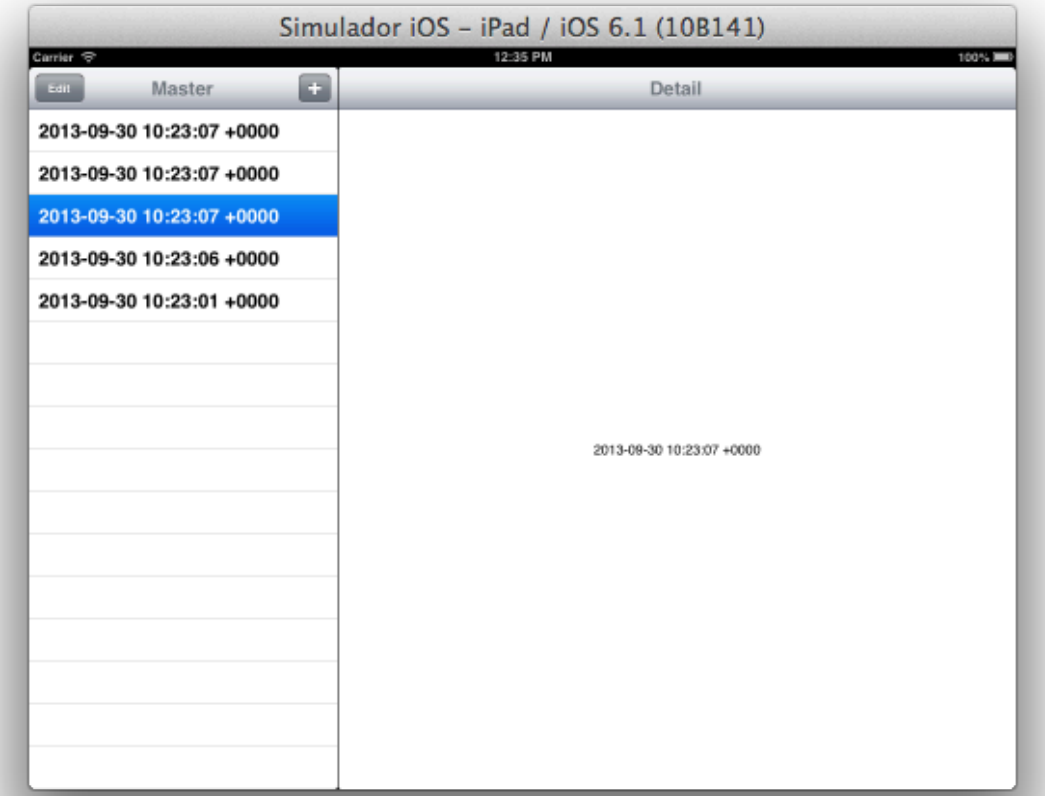
```
// Para poner el botón de la izquierda:
[self.navigationItem setLeftBarButtonItem:barButtonItem
                                 animated:YES];

// Para eliminar el botón de la izquierda:
[self.navigationItem setLeftBarButtonItem:nil
                                 animated:YES];
```

- En una ToolBar

```
// Array mutable es una copia de los items de la ToolBar:
NSMutableArray * array = [self.mytoolbar.items mutableCopy];

// Para eliminar algun elemento:
[array removeObject:barButtonItem];

// Para añadir algun elemento:
[array insertObject:barButtonItem atIndex:0];

// actualizo la ToolBar con los nuevos items:
self.mytoolbar.items = array;
```

Demo 1
iPad: Plantilla Master-Details

- Xcode proporciona una plantilla para crear nuevos proyectos llamada **Master-Detail Application**.
  - Si el proyecto es para **iPad**, esta plantilla crea un **Split View Controller**.



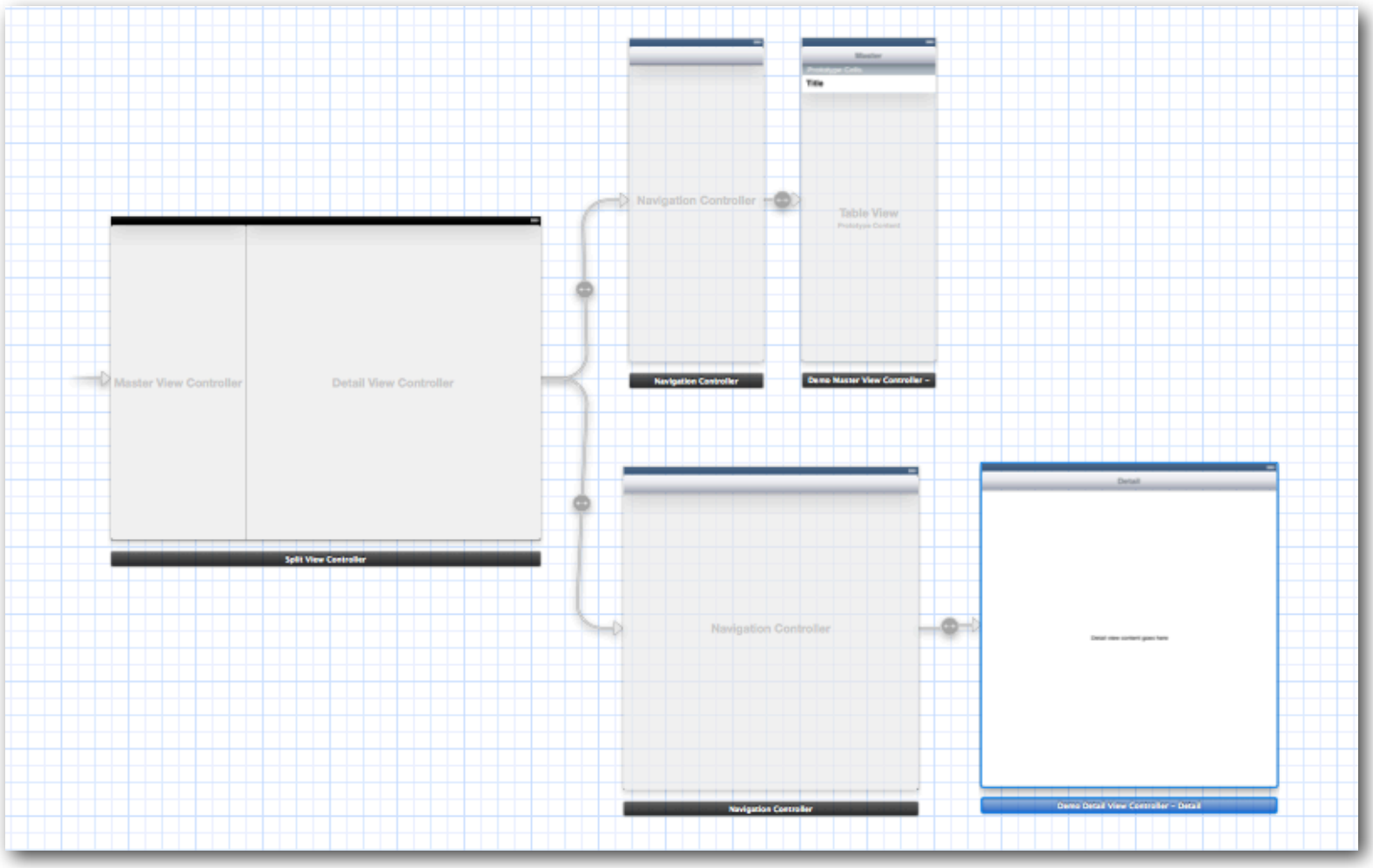
- Pasos de la demo:

1. Crear un proyecto para iPad basado en la plantilla Master-Detail.

2. Examinar el contenido de los ficheros:
   2.1. Storyboard
   2.2. AppDelegate
   2.3. MasterViewController
   2.4. DetailViewController

Notas

El storyboard muestra un SplitViewController que apunta a MasterViewController y a DetailViewController.
Ambos embebidos en un NavigationViewController.



Notas

En **`AppDelegate.m`**

- En **`application:didFinishLaunchingWithOptions:`**
  - Se pone el DetailVC como el delegado del SplitViewController.

En **`MasterViewController`**

- Esta clase tiene una propiedad que apunta al DetailVC.
  - Se asigna su valor en viewDidLoad
- Esta clase es un TableVC
  - Al seleccionar una entrada de la tabla se actualiza el contenido de DetailVC.
    - No se destruye el objeto DetailVC para sustituirlo por objeto DetailVC nuevo.
      - Esto sería un problema ya que el objeto DetailVC es el delegado del SplitViewController.

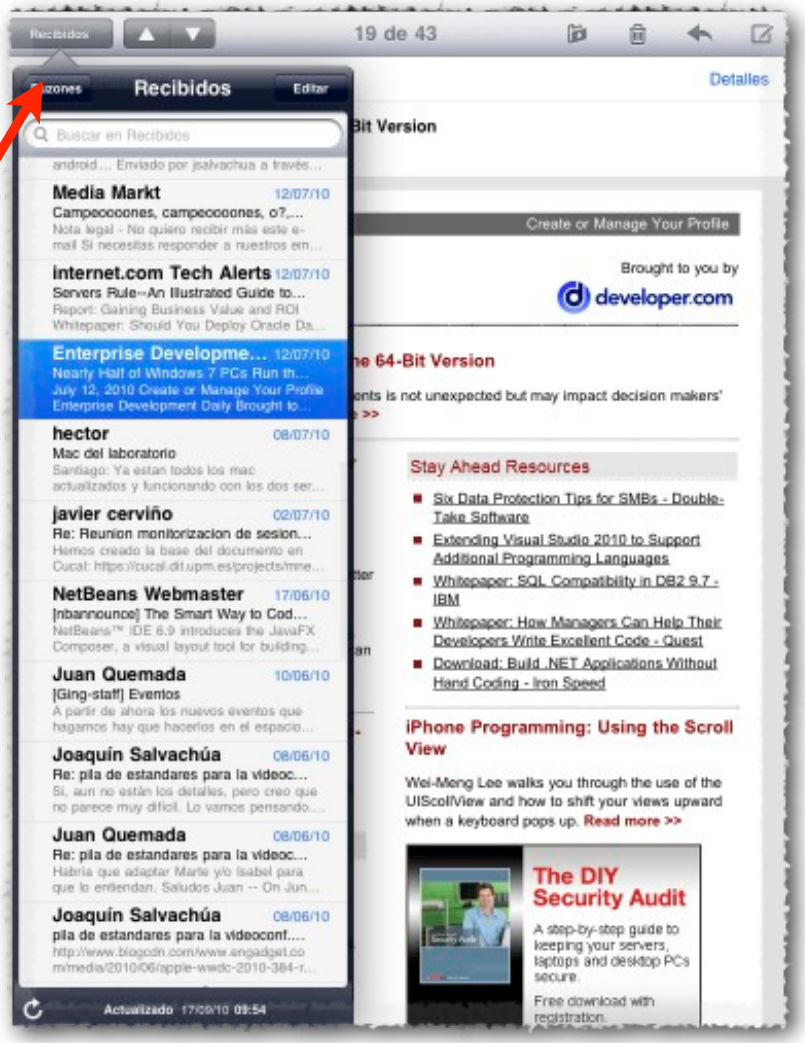
En **`DetailViewController`**

- Tiene una propiedad (detailItem) que apunta al contenido a mostrar.
- Tiene la implementación de los métodos del protocolo delegado UISplitViewControllerDelegate.
- Y tiene una propiedad que apunta al PopoverController que muestra el MasterVC.
  - Se usa para ocultar el popover al actualizar los detalles.

Popover Controller

Popover Controller

- Permite sacar en la pantalla una ventana que muestra un UIViewController.
- La ventana tiene una marca que apunta al elemento que la hizo aparecer.



Creación

- Programáticamente:
  - Pedimos memoria con `alloc` e inicializamos con:

```
-(id)initWithContentViewController:(UIViewController*)vc
```

- Gráficamente:
  - Crear un segue de tipo **popover**
    - Ctrl+Arrastrar sobre un VC.
    - Al dispararse ese segue se muestra el VC destino en un `UIPopoverController`.
  - Configurar la transición adaptando (en el VC origen) el método

```
-(void)prepareForSegue:(UIStoryboardSegue *)segue
                sender:(id)sender
```

    - El parámetro segue es de tipo **`UIStoryboardPopoverSegue`**
      - Deriva de **`UIStoryboardPopoverSegue`** añadiendo una propiedad adicional:

```
UIPopoverController* popoverController
```

        - apunta al `UIPopoverController` creado para mostrar el VC destino.
    - Recordad que hay que asignar un identificador único al segue creado
      - `prepareForSegue:sender:` lo usa para identificar cual ha sido el segue disparado.

Propiedades

- VC mostrado:
  ```
  UIViewController * contentViewController
  ```
- Tamaño de la view del VC mostrado:
  ```
  CGSize popoverContentSize
  ```
- Views que puedo tocar sin que desaparezca el popover:
  ```
  NSArray * passthroughViews
  ```
- El delegado:
  ```
  id<UIPopoverControllerDelegate> delegate
  ```
- Indicar si el popover está visible:
  ```
  BOOL popoverVisible
  ```
- Dirección de la flecha indicadora usada por el popover:
  ```
  UIPopoverArrowDirection popoverArrowDirection
  ```
- Apariencia:
  ```
  UIColor* backgroundColor
  UIEdgeInsets popoverLayoutMargins
  Class popoverBackgroundViewClass
  ```
- ...

Métodos

- Contenido:

```
- (void)setContentViewController:(UIViewController *)vc
                        animated:(BOOL)animated

- (void)setPopoverContentSize:(CGSize)size
                     animated:(BOOL)animated
```

- Presentar:

```
- (void)presentPopoverFromRect:(CGRect)rect
                        inView:(UIView *)view
      permittedArrowDirections:(UIPopoverArrowDirection)directions
                      animated:(BOOL)animated;
- (void)presentPopoverFromBarButtonItem:(UIBarButtonItem *)item
               permittedArrowDirections:(UIPopoverArrowDirection)dir
                               animated:(BOOL)animated;
```

- Ocultar:

```
- (void)dismissPopoverAnimated:(BOOL)animated;
```

Retener Popover Controller

- Los objetos `UIPopoverController` deben retenerse con alguna variable strong.
  - No es suficiente con presentar el popover.
- Esto es incorrecto (*la app se muere al intentar liberar la memoria de un popover visible*):

```
{  UIPopoverController * pop = [[UIPopoverController alloc]
                                    initWithContentViewController:vc;
   [pop presentPopoverFromRect:rect   inView:v
      permittedArrowDirections:dirs animated:YES];
}
```

- Esto es correcto:

```
@property (nonatomic, strong) UIPopoverController *pop; // retiene

. . .

self.pop = [[UIPopoverController alloc]
                        initWithContentViewController:vc];
[self.pop presentPopoverFromRect:rect   inView:v
       permittedArrowDirections:dirs animated:YES];
```

UIPopoverControllerDelegate

- Se invoca cuando se quita el popover.
  - no se llama si el popover se quita programáticamente.

```
- (void)popoverControllerDidDismissPopover:
            (UIPopoverController *)popoverController
```

- Pregunta si se permite quitar el popover:

```
- (BOOL)popoverControllerShouldDismissPopover:
            (UIPopoverController *)popoverController
```

Demo
Crear un Popover Controller (con Storyboard)

Pasos

- Crear un proyecto Single View Application para iPad.
- Editar el storyboard:
  - Añadir un botón al VC existente.
  - Añadir un nuevo VC al Storyboard.
    - Poner una label en este segundo VC.
  - Crear un segue de tipo Popover desde el botón del primer VC al segundo VC.
    - Ctrl+Arrastar desde el botón hasta el 2º VC.
- Ya puede ejecutarse esta aplicación.
- Más pasos:
  - Añadir un identificador al segue.
    - Y modificar el método `prepareForSegue:sender:` del primer VC.
  - Cambiar alguna propiedad del 2º VC en el inspector:
    - Por ejemplo, cambiar el tamaño editando el valor de Popover - Use Explicit Size