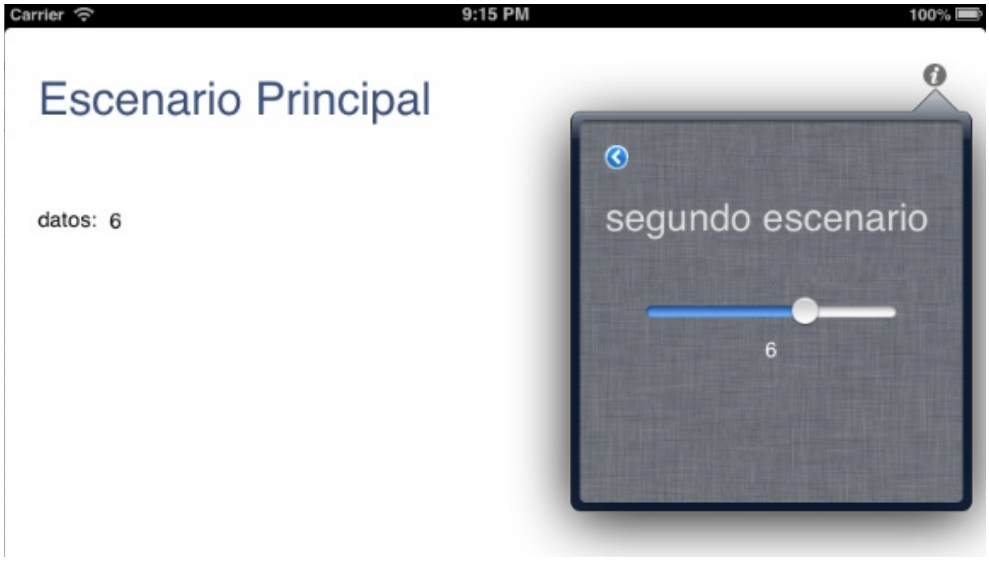
protocolos y delegates (popover)

los protocolos para delegates definen a un controlador para que trabaje para otro controlador. esto es útil cuando las acciones de un controlador afectan a otro y viceversa. en estos casos siempre hay un controlador "amo" y uno "esclavo", el controlador amo generalmente es el principal y el encargado de crear el segundo controlador; el escalvo. por supuesto que el esclavo también podría ser un objeto general como un modelo de datos, sin embargo, tiene más sentido entre controladores pues en el paradigma MVC cuando un controlador le pasa el control al próximo deja de existir y por lo tanto no podrían comunicarse entre si.

en el siguiente ejemplo se crea un controlador para crear una ventana del tipo popover que contiene un slider que cambia una propiedad en el controlador principal:



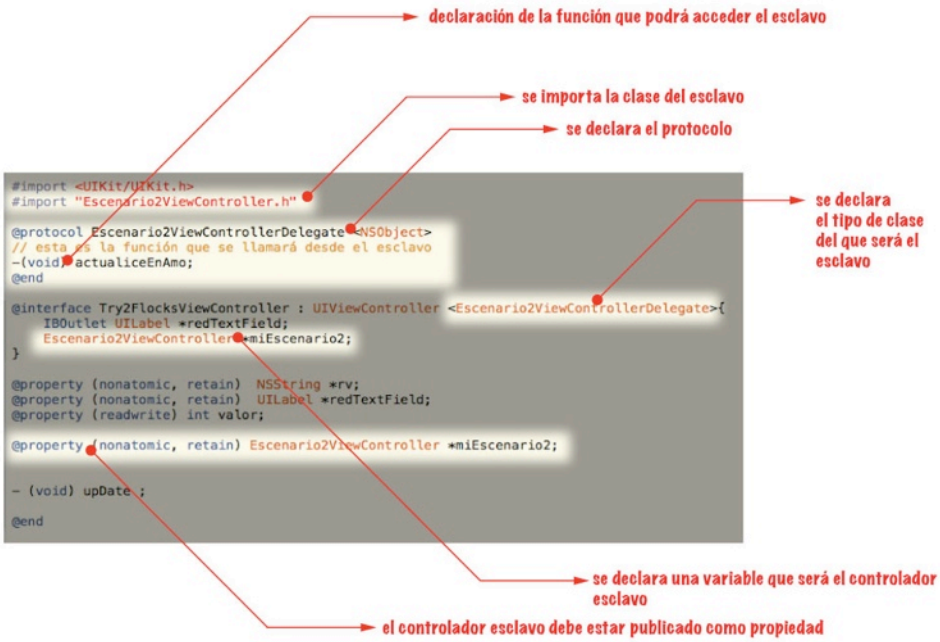
para lograr esta relación se debe:

1. declarar cada objeto en el otro controlador
2. declarar que van a ser usados como delegate y amo
3. declarar las funciones que se van a usar desde el otro controlador en cada uno de ellos

en el controlador amo:

.h define

en primer lugar se el tipo de controlador que se va a usar como esclavo en el controlador principal, específicamente en .h:



como se ve, en el .h del controlador principal es necesario:

1. se importa la clase del esclavo
2. se declara en protocolo
3. se declara la función que podrá usar el esclavo
4. se declara la clase del esclavo como clase a usar como delegate
5. se crea una variable del objeto del esclavo
6. se convierte la variable del esclavo en propiedad del amo

.m

en el .m del controlador principal (amo) lo primero es sintetizar la propiedad en la que se accede al controlador esclavo:

```
#import "Try2FlocksViewController.h"

@implementation Try2FlocksViewController

@synthesize rv, redTextField, valor;
@synthesize miEscenario2;


- (void) upDate {
    printf("Inside Try2FlocksViewController | upDate : valor = %d \n", valor);
    rv =[[NSString alloc] initWithFormat:@"%d",valor];
    [redTextField setText:rv];
}
```

lo primero a hacer es crear el controlador esclavo, en este caso el controlador servirá para generar una ventana popover así que este controlador se crea cada vez que se presiona el botón que conecta ambos controladores. así que el prepareForSegue es el mejor lugar para crearlo:

```
-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender{
    if([[segue identifier] isEqualToString:@"HaciaEscenario2"]){
        miEscenario2 = (Escenario2ViewController *)[segue destinationViewController];
        miEscenario2.delegate = self;
        printf("desde segue en escenario principal \n");
        [self.miEscenario2 actualiceEnEsclavo];
    }
}
```

nótese que aquí mismo se declara inmediatamente después de creado el controlador nuevo (esclavo) como el delegate de este "amo". además se llama a la función que el amo puede llamar del esclavo, esto es solo como ejemplo.

***

el siguiente paso en este mismo .m es definir la función que puede ser accesible desde el esclavo:

```
#pragma para protocolo ------------------------------------------------------

// esta es la función que se llamará desde el esclavo
-(void) actualiceEnAmo{
    printf("desde actualiceCantidadEnAmo \n");
    valor=miEscenario2.redval;
    [self upDate];
}

@end
```

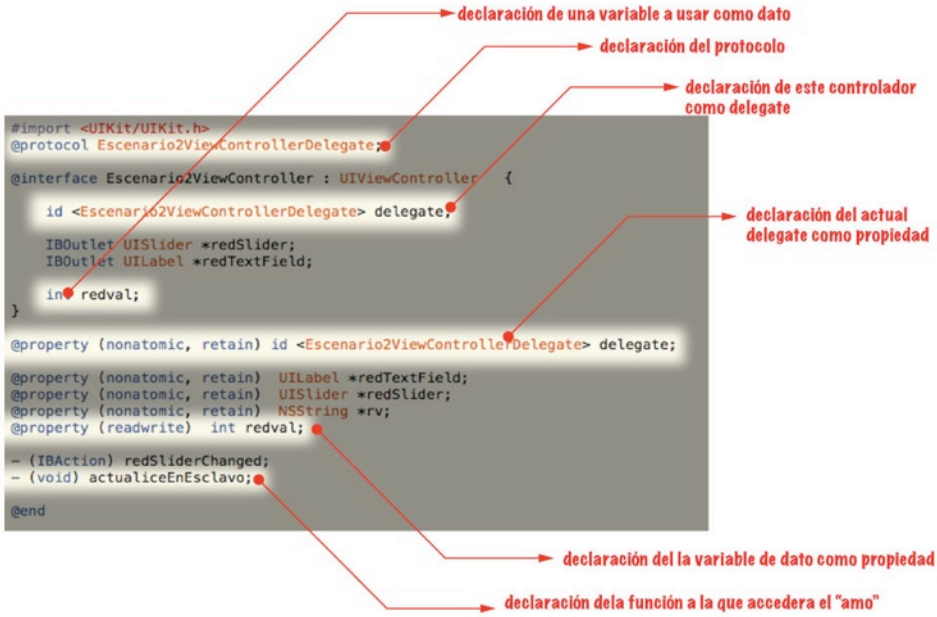
en este caso se puede ver como la propiedad "valor" en el amo se iguala con la propiedad redval en el esclavo, típica situación de sincronización entre ambos controladores, después se actualiza el amo, esto es útil en caso de que hayan cosas que hacer en el amo con el nuevo valor cambiado en el esclavo.

en el controlador esclavo:

.h

en el controlador esclavo .h se deben hacer varias acciones:

1. declara el protocolo
2. declaración de este controlador como delegate
3. creación de una variable para ser usad de dato entre controladores
4. declaración del actual delegate como propiedad del controlador esclavo
5. declaración de la función que será accedida desde el controlador amo



.m

en el .m del controlador esclavo lo primero que se hace es importar la clase del controlador amo, esto es necesario pues para cada acción con este será necesario crear momentáneamente punteros locales al amo.

```
#import "Escenario2ViewController.h"
#import "Try2FlocksViewController.h"


@implementation Escenario2ViewController

@synthesize redTextField, redSlider, rv;
@synthesize delegate, redval;
```

además en este mismo paso se sintetiza la variable de datos.

el siguiente paso es preparar el segue, como se ve en la figura esto se hace creando momentáneamente una variable para punterala con el amo, y de paso se sincronizan los datos entre los dos controladores:

```
-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender{
    if([[segue identifier] isEqualToString:@"DesdeEscenario2"]){
        Try2FlocksViewController *cvc = (Try2FlocksViewController *)[segue destinationViewController];
        cvc.valor = redval;
        [[NSUserDefaults standardUserDefaults] synchronize];
        printf("Inside prepareForSegue = %d \n", redval);
    }
}
```

en la acción del elemento interactivo que cambia el usuario (en este caso un slider), se puede llamar la función del amo para actualizar inmediatamente la visualización en el controlador principal:

```
- (IBAction) redSliderChanged {
    redval = (int)(redSlider.value + 0.5f);
    rv =[[NSString alloc] initWithFormat:@"%d",redval];
    [redTextField setText:rv];
    // Store the data
    NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
    [defaults setInteger:redval forKey:@"catidadDeTry2Flocks"];
    [delegate actualiceEnAmo];
}
```

finalmente se debe implementar la función que será llamada por el amo en el esclavo;

```
#pragma para protocolo ----------------------------------------------------------------------
-(void) actualiceEnEsclavo {
    printf("en actualiceCantidad dentro del esclavo ------------------\n");
}
```