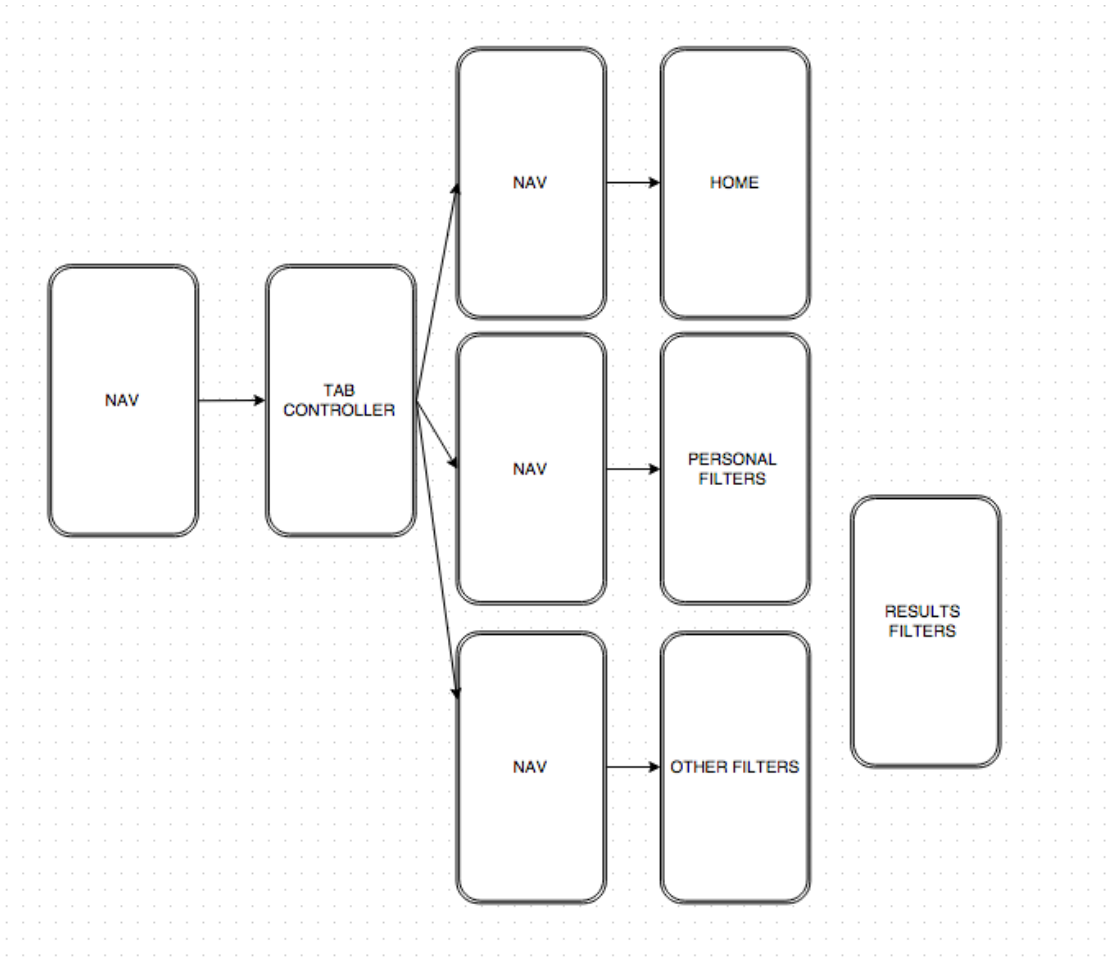
Memoria técnica de la propuesta de candidatos de AXA.

1.- Introducción

En este documento se recoge toda la información del proyecto de la propuesta de candidatos realizada por Marc Gallardo Ruiz. El documento esta dividido en secciones, donde cada sección se explica el funcionamiento por pantallas.

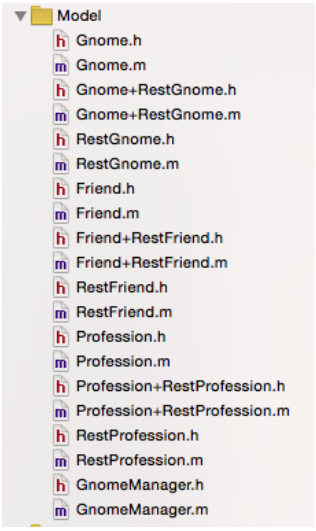
2.- Diagrama de la app

Este es el diagrama de la aplicación, se ha simplificado la navegación para el usuario situando al principio un tab bar controller.

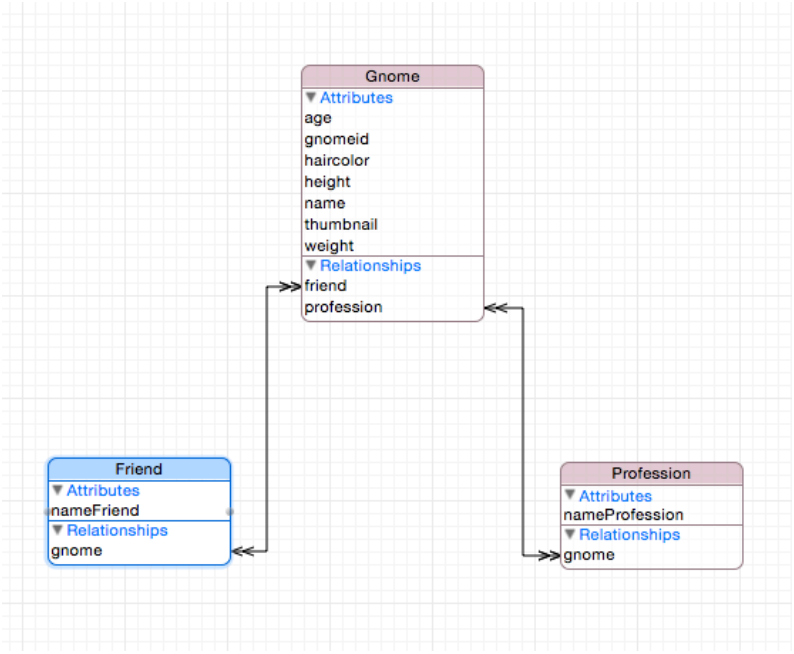


3.- Modelo de Datos

Para la gestión de todos los datos se ha utilizado la siguiente arquitectura:



Además de un modelo de datos en Core Data:



Dentro de la carpeta **Model** se han añadido los siguientes archivos**: Gnome, Friend y Profession**, que representan las identidades del modelo de datos CoreData, se encargan de almacenar los datos que vienen procedentes del adapter **Gnome+RestGnome**. Los archivos **Rest** son los encargados de gestionar los datos que vienen procedentes de las peticiones **JSON** que realiza el **GnomeManager.** Finalmente el adapter comentado a priori es el encargado de gestionar los datos de Core Data ha JSON y de JSON a Core Data.

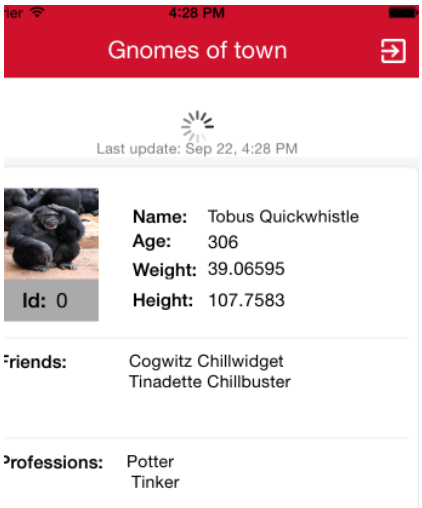
4.- Home

Cuando se cargan los datos estos son presentados en una **table view controller** de inicio, al inicio se detecta que el **array** de la tabla no este vacía, si esta vacía carga una función desde GnomeManager que se encarga de hacer la solicitud de los datos al JSON.

```
        if([self.gnomes count] == 0){
        MBProgressHUD *hud = [MBProgressHUD HUDForView:self.view];
        if (!hud) {
            hud = [[MBProgressHUD alloc]initWithView:self.view];
            [self.view addSubview:hud];
        }

        [hud setLabelText:@"Loading..."];
        [hud setMode:MBProgressHUDModeIndeterminate];
        //[hud setAnimationType:MBProgressHUDAnimationZoom];

        [hud show:YES];

        [[GnomeManager sharedManager] attemptGetGnomesWithSuccess:^(NSArray *gnomes) {
            self.gnomes = [Gnome MR_findAll];

            [self.tableView reloadData];


            [[MBProgressHUD HUDForView:self.view] hide:YES];
        }];
        }
    }
```

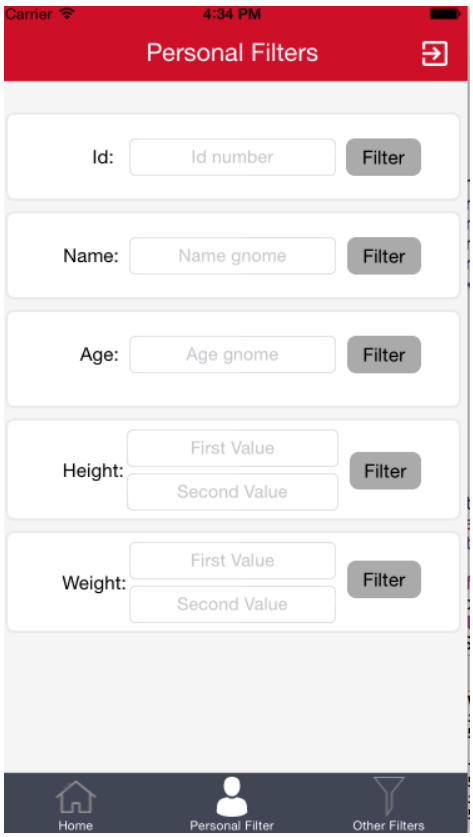
En el caso que el array este lleno se ha programado un refresh control, permite al usuario solicitar un refresh de los datos llamando a la función que le permite descargar los datos del **JSON**. Para facilitar una buena **UX**.

```
self.refreshControl.attributedTitle = attributedTitle;

[[GnomeManager sharedManager] attemptGetGnomesWithSuccess:^(NSArray *
    gnomes) {
 self.gnomes = [Gnome MR_findAll];
 [self.tableView reloadData];
 [self.refreshControl endRefreshing];
 }];
```



5.- Personal Filters

Cuando iniciamos esta pantalla se nos presenta varias formas de filtrar la información, a través de varios campos podemos aplicar un filtro ya sea **(Nombre, Id, Edad, Peso y Altura).**



Para poder hacer el filtrado de información se han utilizado las funciones que dispone **Magical Record** y con pasarle una string ya detecta si hay algún campo con el mismo nombre en la entidad.

```
Gnome *gnome = [Gnome MR_findFirstByAttribute:@"gnomeid"
                                    withValue:self.idTextField.text];
[self checkgnome:gnome textgnome:self.idTextField];
```

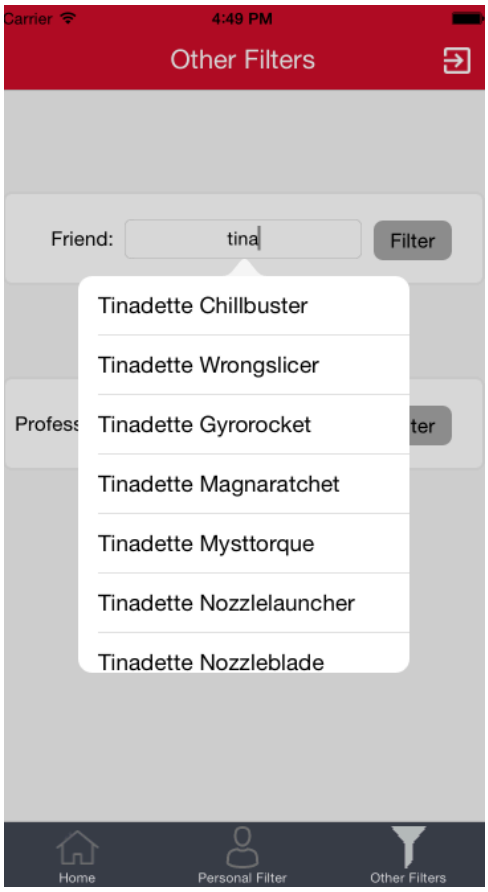
Para la altura y el peso se ha modificado la búsqueda añadiendo un NSPredicate, para así poder hacer la búsqueda dentro de un rango.

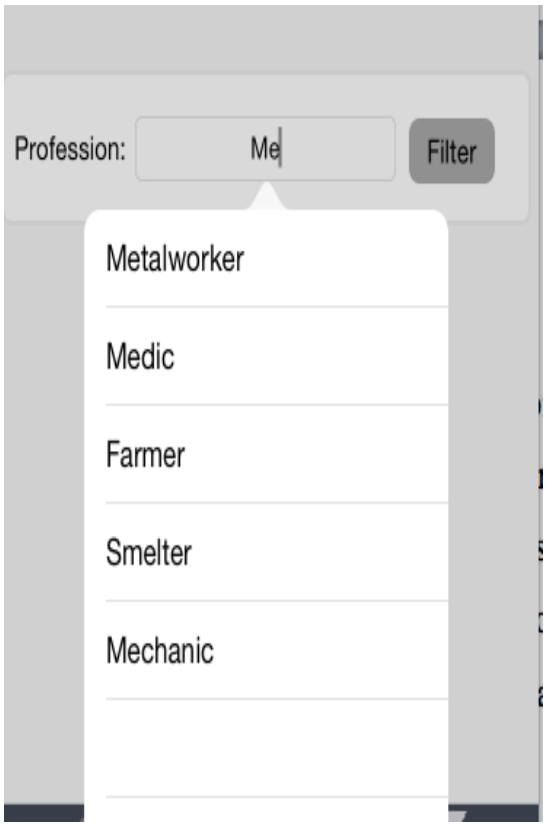
```
NSPredicate *peopleFilter2 = [NSPredicate predicateWithFormat: @"weight > %@
    && weight < %@",self.weight1TextField.text,self.weight2TextField.text];

NSArray *people2 = [Gnome MR_findAllWithPredicate:peopleFilter2];
```

Para presentar los resultados se ha añadido el mismo tipo de celda utilizado en la **HOME**.

6.- Other Filters

He separado los filtros personales como pueden ser números o palabras que podemos cargar in situ de los que es los nombres de los amigos y profesiones, cuando se carga esta pantalla podemos observar que sigue el mismo planteamiento que la anterior pero hay una diferencia, cuando el usuario empieza escribir en cualquier de los dos campos, la app hace una búsqueda según en las letras que esta poniendo el usuario y presenta una lista con todas las coincidencias en un **popover**.





Para poder realizar esta acción se ha contado con una clase externa llamada **ARSPopover**, que te permite añadir una popover a un dispositivo **Iphone** además de añadir un table view dentro del pop over. Para poder rellenar la tabla primero se ha creado un array de todos los nombres de amigos y profesiones por separado, una vez tenemos la array se ha realizado un filtrado del array con **NSPredicate** y se ha utilizado el array filtrado para rellenar la tabla.

7.-Bibliografía

ARSPopover: https://github.com/soberman/ARSPopover

AFNetowrking: https://github.com/AFNetworking/AFNetworking

SDWebimage: https://github.com/rs/SDWebImage

MagicalRecords: https://github.com/magicalpanda/MagicalRecord

MBProgressHUB: https://github.com/jdg/MBProgressHUD