Práctica de Fundamentos iOS

Teoría complementaria

Gestión de memoria en iOS

Algo que a menudo se pasa por alto, es que Swift (y Objective C) no tienen un recolector de basura, al contrario de otros lenguajes como Kotlin o Javascript.

Tienen algo parecido, llamado ARC (Automatic Reference Counting).

Explicación en video

1. ¿Qué es ARC y cómo funciona?
2. ¿Qué es una referencia circular fuerte y cómo podemos evitarla?

Explicación en texto

1. ¿Qué es ARC y cómo funciona?
2. ¿Qué es una referencia circular fuerte y cómo podemos evitarla?

Obligatorio

1

- Crea la clase `Season` que representa una temporada. Debe de tener un número > 0 de episodios, un nombre y fecha de lanzamiento (ver `Date`).
- Cada episodio está representado por una instancia de `Episode`. Cada `Episode` tiene un título y una fecha de emisión.

Todo episodio tiene una referencia a la `Season` de la que es parte. ¡Mucho cuidado con crear referencias circulares! Échale un ojo a ARC y las referencias circulares en la parte de teoría. (SPOILER: La referencia que tiene `Episode` a `Season` ha de ser `weak` 😉)

2

Implementa los siguientes **protocolos** en `Episode` y `Season` **con sus correspondientes tests unitarios**:

- `CustomStringConvertible`
- `Equatable`
- `Hashable`
- `Comparable`

IMPORTANTE: ¡No te olvides de escribir los tests!

3

Crea una propiedad calculada en tu `Repository.local`, llamada `seasons` (similar a `houses`). Devuelve las primeras 7 temporadas, ordenadas, con sus episodios (2 por temporada, no hace falta ponerlos todos 😆).

4

Crea una función en tu `Repository.local` llamada `seasons(filteredBy:) -> [Season]` que acepta una clausura (similar a lo que hicimos para houses) y te permite recibir un `[Season]` filtrado.

5

Crea un `SeasonListViewController`. Al hacer clic sobre una celda, se debe hacer un push de un `SeasonDetailViewController`.

6

Crea un `EpisodeListViewController`. Al hacer clic sobre una celda, se debe de hacer un push de un `EpisodeDetailViewController`.

Podéis ser creativos y añadir las propiedades que queráis a los modelos. Ejemplo: añadir resumen a `Episode` y mostrarlo en el `EpisodeDetailViewController`.

7

Nueva Interfaz gráfica:

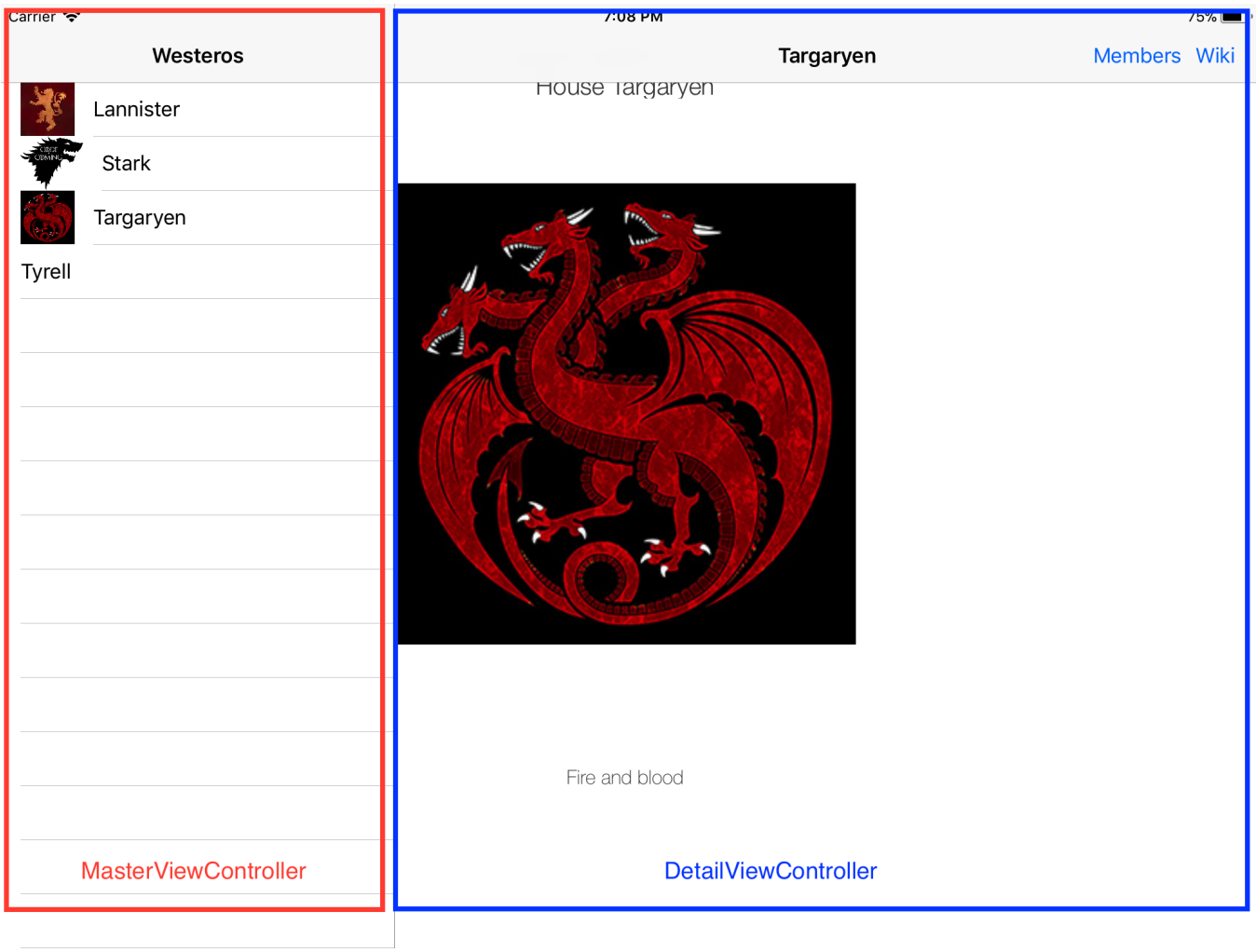
- Crea un `HouseListViewController` empaquetado dentro de un `UINavigationController`.
- Crea un `SeasonListViewController` empaquetado dentro de un `UINavigationViewController`
- Mete a ambos dentro de un `UITabBarController`
- Usa éste como `masterViewController` del `UISplitViewController`. Asegúrate de poder cambiar de uno a otro y poder navegar de `House` a `Person` y de `Season` a `Episode`.

Pista: `UITabBarControllerDelegate`
Pista 2: Guarda como propiedades todos aquellos objetos que necesites utilizar en varios métodos de la misma clase.

8

- Crea una función `house(:named)`, similar a la que acepta un `String`, pero que sea type safe y funcione el autocompletado. ¿Se te ocurre qué tipo de datos podemos utilizar? Haz el test correspondiente.

9



Cuando estamos en un iPad, y en el `detailViewController` se muestra el `WikiViewController`, al seleccionar otra casa en `masterViewController`, se muestra el `WikiViewController` de la nueva casa seleccionada.

Sin embargo, cuando se muestra `MembersViewController`, esto no ocurre. **Arréglalo** 🛠️.

Si has hecho el ejercicio 7:
Asegúrate que, con la nueva interfaz, al seleccionar otra `Season` se actualiza su detalle y su lista de capítulos.

10

Crea un `MemberDetailViewController`. Cuando se haga click en una celda de `MemberListViewController`, hacer un push a `MemberDetailViewController`.

Opcional

11

En nuestra clase `Repository`, cuando creamos un personaje, le pasamos por parámetro la casa a la que pertenece:

```
1 arya = Person(name: "Arya", house: starkHouse)
```

Sin embargo, para que el personaje pertenezca a la casa debemos añadirlo manualmente:

```
1 starkHouse.add(person: arya)
```

¿No os resulta esto un poco redundante? ¿Cómo podríamos hacer para que al crear un nuevo personaje y asignarle la casa, se actualice su propiedad `_members` 🤔? Impleméntalo.

> > En el siguiente módulo veréis que, utilizando `CoreData`, se actualiza automágicamente

12

Cuando utilizamos el `UISplitViewController` en iPhone, nos damos cuenta de varias cosas: por una parte, el `UISplitViewController` parece haberse transformado en un `UINavigationController`; por otra, parece que ya se ha hecho un *push* a `HouseDetailViewController`.

Cuando volvemos hacia atrás y seleccionamos de nuevo algún `House`, no se produce la navegación. Arréglalo 🛠.