UI Router

¿Qué es?

- Es un **Framework** para Angular desarrollado por el equipo de **AngularUI**
- Proporciona un enfoque basado no solo en rutas sino en estados de la aplicación

Estados Vs Router

- Permite definir estados, y la transición de la aplicación a esos estados
- Las vistas y rutas ya no están necesariamente unidas a la URL

Estados Vs Router

- Permite hacer diseños muy complicados de una manera elegante
- La verdadera ventaja es poder tener vistas anidadas

angular-ui-router.js

- Como con ngRoute, necesitamos instalarlo por separado y por supuesto incluirlo en nuestra aplicación

```
var app = angular.module(app, ['ui.router']);
```

$urlRouterProvider

- Es como el $routeProvider de ngRoute
- Cuando $location cambia, ejecuta una serie de reglas una a una hasta que alguna coincide

$urlRouterProvider

```
app.config(['$stateProvider', '$urlRouterProvider',
function($stateProvider, $urlRouterProvider) {
    $urlRouterProvider.otherwise('/');
                    .....
```

$stateProvider

- Un **estado** corresponde a un "**lugar**" de la aplicación en cuanto a la navegación de la UI se refiere

$stateProvider

- Describe a través del controlador, la plantilla y las propiedades de la vista, cuál será la interfaz de usuario y el comportamiento de esta en ese estado

$stateProvider

```
$stateProvider
    .state(uilogic, {
        url:'/',
        templateUrl: 'modules/uilogic/views/uilogic.html',
        controller: 'gnaUIlogicCtrl'
    })
```

$stateProvider

- Cuando se activa un estado, su plantilla se inserta automáticamente en el **ui-view**

```
<body ng-app="gnasApp">
        <div id="navbar" navbar></div>
        <div id="content" ui-view></div>
```

$stateProvider: template

- Podemos configurar la plantilla del estado
  - Con “**template**”

```
$stateProvider.state(promise, {
  template: '<promise-component><promise-component>'
})
```

$stateProvider: template

- Podemos configurar la plantilla del estado
  - Con “**templateUrl**”

```
$stateProvider.state(promise, {
  templateUrl: 'modules/promise/views/promise.html'
})
```

$stateProvider: template

- Podemos configurar la plantilla del estado
  - Con “**templateUrl**” **como función**. Recibe el parámetro preestablecido **$stateParams**, el cual no se inyecta

$stateProvider: template

- “**templateUrl**” como función

```
$stateProvider.state(promise, {
    templateUrl: function ($stateParams){
        return '/module/promise/views.'+
            $stateParams.filterBy + '.html'; }
})
```

$stateProvider: template

- Podemos configurar la plantilla del estado
  - Con la función “**templateProvider**”. Es inyectada, y tiene acceso a las variables locales. Devuelve una plantilla html

$stateProvider: template

- “templateProvider”

```
$stateProvider.state(promise, {
    templateProvider: function ($timeout, $stateParams) {
        return $timeout(function () { return 'promise.html'; }
        }, 100);
    }
})
```

$stateProvider: controller

- También podemos asignar controladores al template de varias maneras
- Definiéndolo directamente en el estado, instanciando uno existente, con "**controllerProvider**", etc

$stateProvider:activar

- Hay 3 maneras de activar un estado
  - Llamando `$state.go()`
  - Clicando en un link que contiene `ui-sref`
  - Navegando a la `url` asociada al estado

$stateProvider:eventos

- **$stateChangeStart**
  - Detonado cuando la transición empieza

```
$rootScope.$on('$stateChangeStart',
function(event, toState, toParams, fromState, fromParams){ ... })
```

$stateProvider:eventos

- **$stateChangeStart**
  - Podemos prevenir que la transición se efectúe

```
event.preventDefault()
```

$stateProvider:eventos

- **$stateChangeSuccess**
  - Detonado cuando la transición termina

```
$rootScope.$on('$stateChangeSuccess',
function(event, toState, toParams, fromState, fromParams){ ... })
```

$stateProvider:eventos

- **$stateChangeError**
  - Si se hay error durante la transición

```
$rootScope.$on('$stateChangeError,
function(event, toState, toParams, fromState, fromParams){ ... })
```

Ejercicios

- Sustituye el actual modo de enrutamiento de ngRoute por el de Angular UI
  - Cambia el **$routerProvider** por **$urlRouterProvider** y **$stateProvider**

Ejercicios

- Sustituye el actual modo de enrutamiento de ngRoute por el de Angular UI
  - Arregla la directiva **navbar** (utiliza **$stateChangeSuccess)**

Ejercicios

- ¿Siguen funcionando los tests?