Diseño de Software para Dispositivos Móviles

José Ladislao Lainez Ortega y José Molina Colmenero

11 de mayo de 2013

Índice

1. Introducción

El siguiente documento muetra la información relativa al diseño e implementación de una aplicación de gestión de tareas para iOS, como parte de la asignatura Diseño de Software para Dispositivos Móviles del Grado de Ingeniería informática de la Universidad de Jaén.

2. Concepto

Para dar algo de originalidad al proyecto vamos a usar una aproximación distinta al sistema de tareas tradicional, de forma que podamos aplicar esta nueva filosofía en el diseño de la aplicación desde el principio.

2.1. Getting Things Done

O comúnmente conocida como GTD es una metodología de organización de tareas enfocada a la productividad. Sus pilares clave son la sencillez y la categorización, de forma que el usuario no tiene que aprender complejos patrones ni adaptar drásticamente su forma de organizarse.

El funcionamiento base de GTD persigue que el usuario se olvide de las tareas hasta el momento en que tiene que hacerlas, de forma que pueda concentrarse en aquellas tareas que están más cerca en el tiempo. Para ello se hace uso de varias categorías que clasificarán las tareas según unos criterios de cercanía en el tiempo y que serán revisadas una vez al día.

Para conseguir que el usuario no tengan en mente las tareas que tiene que hacer y se pase el día pensando "Que no se me olvide esto, cuando llegue a casa tengo que hacerlo"GTD propone que la creación de tareas sea extremadamente sencilla. La idea consiste en crear las tareas justo en el momento en que las piensas, pero solo especificando un título para esta; no será necesario especificar categoría, fecha ni ningún otro parámetro, el objetivo en ese momento es que el usuario cree la tarea y se olvide de ella para que no siga dando vueltas en su cabeza, ya llegará el momento de atenderla. Estas tareas recién creadas van a la categoría llamada 'Inbox'.

A lo largo del día el usuario habrá creado varias tareas que ahora están en la categoría 'Inbox'. ¿Qué hace con estas tareas? Pues buscar un momento al día en el que revisar las tareas que se encuentran en 'Inbox'y asignarle una nueva categoría aparte de proporcionar la información adicional que sea necesaria.

En este punto hay varias versiones de GTD que usan categorías distintas pero nosotros nos quedaremos en la más básica para mantener la aplicación lo más sencilla de usar, por

lo que tendremos las siguientes categorías para asignar a nuestras tareas del 'Inbox'según corresponda a cada una:

Next
: Aquellas tareas que se pueden realizar próximamente. "Comprar el pan".

Waiting
: Tareas que podrían pertenecer a 'Next'pero dependemos de alguien o algo para poder realizarlas. "Ir de picnic un día soleado".

Project
: Relacionadas con algún proyecto en el que el usuario está involucrado. "Corregir el fallo en la interfaz de la aplicación para DSDM".

Someday
: Cosas que se quieren realizar algún día, planes a largo plazo. "Comprar un MacBook Pro Retina Display".

De esta forma hemos dividido en dos etapas el flujo de trabajo; durante la primera se crean las tareas, que quedarán almacenadas en un buzón ('Inbox') para que no tengamos que pensar en ellas durante todo el día y la segunda en la que revisamos las tareas para actualizar su información ya sea cambiándolas de categoría, marcándolas como completadas o eliminándolas si ya no son necesarias.

3. Diseño

Siguiendo la filosofía anterior diseñaremos nuestra aplicación para que se amolde al esquema de funcionamiento de GTD, procurando que sea lo más sencillo posible.

3.1. Tareas

Las tareas estarán compuestas por los siguientes atributos:

- Nombre.
- Descripcion o notas.
- Prioridad.
- Fecha de creación.
- Categoría.

Al crear la tarea esta recibirá la categoría 'Inboxáutomáticamente, pudiendo ser esta modificada más adelante durante la revisión de las tareas. Los atributos 'prioridad'y 'fecha de creación'se han añadido por conveniencia a pesar de no ser necesarios en la metodología GTD para aportar más opciones de ordenación a la hora de visualizar el listado de tareas de una categoría.

3.2. Categorías

Ya hemos comentado antes una serie de categorías relacionadas con GTD, pero hemos de añadir otras dos más para contemplar dos posibles estados de las tareas; completadas y borradas.

- Inbox.
- Next.
- Waiting.
- Project.
- Someday.
- Completed.
- Trash.

Las categorías 'Completed'y 'Trash'son asignadas a las tareas completadas y eliminadas respectivamente, de forma que podamos listar estas tareas en nuestra aplicación. Aunque estas categorías no entren dentro de GTD eran necesarías por los requisitos impuestos para el proyecto.

3.3. Interfaz

La funcionalidad que debe otorgar la aplicación debe ser directa y sencilla para que la creación de tareas sea lo más ágil posible y debemos poder ver las tareas por categorías a la hora de revisarlas y editarlas durante la revisión diaria de tareas.

Por tanto la aplicación debe contar con las siguientes vistas:

Home
: Acceso directo al 'Inbox'y a la creación de tareas, así como a las categorías.

Creación de tarea
: Permite crear una tarea nueva.

Categoría
: Listado de las tareas pertenencientes a una categoría. Debe permitir ordenar la lista según varios criterios.

Tarea
: Visualzación de la información de una tarea concreta. Se debe poder editar la información de la tarea para realizar, por ejemplo, el cambio de categoría. También se debe poder marcar la tarea como completada o borrarla si ya no es necesaria.

Edición
: Edición de la información de la tarea. Se debe poder escoger la categoría a la que asignarla.

3.4. Notificaciones

Aunque en un principio pueda parecer que insertar notificaciones para las tareas individuales que se encuentran en 'Inboxés una buena idea, rompe con un concepto fundamental de GTD: que el usuario deje de pensar en la tarea tras crearla y hasta que revise el 'Inbox'. Una buena idea de notificación sería una que avisara al usuario que tiene tareas sin clasificar en la categoría 'Inbox'cada cierto tiempo (unas 12-16 horas estaría bien), de forma que este no olvide revisar sus tareas al menos una vez al día, pero sin saturarlo.

4. Implementación

Para la implementación de la aplicación se ha seguido el patrón Modelo-Vista-Controlador (MVC) de forma que:

Modelo
: Está constituido por todos los objetos que representan la información. Esto incluye tanto las clases utilizadas por la aplicación durante su ejecución como las creadas únicamente para la realización de la persistencia de datos. Asimismo, toda la funcionalidad de gestión de los datos se lleva a cabo en una clase a parte para poder manejar la información de forma más sencilla, tal y como recomienda Apple. Esta clase engloba toda la funcionalidad requerida para la inserción, borrado, modificación y consulta de datos.

Vista
: Los objetos de vista son aquellos que se muestran al usuario como presentación de los datos, es decir, las distintas interfaces con las que el usuario puede interactuar. Los objetos de vista no poseen en sí capacidad de procesar información, sino que son los objetos controladores los encargados de esta tarea.

Controlador
: Los objetos de controlador son aquellos que procesan la información que el usuario aporta y mediante consultas al modelo, muestran al usuario una respuesta. Los controladores se encargan de manejar la lógica de la aplicación, y serían los objetos que controlan cada una de las vistas.

4.1. Tareas

Las tareas pertenecen al grupo de clases del Modelo, y almacenan todos los datos concernientes a esa entidad. Los objetos tarea se utilizan como fuentes de información para poder mostrar los datos de una vista, de forma que cada vista en la aplicación necesita de al menos un objeto tarea del que poder extraer la información.

4.2. Categorías

Las categorías también forman parte del Modelo, dado que se usan para clasificar y organizar las tareas según ese criterio.

4.3. Interfaz

Todos los objetos que pertenecen a la interfaz de usuario a su vez serán clases de Vista. En la interfaz se muestra de forma clara al usuario la información mediante objetos gráficos con los que pueda interactuar mandando información a los controladores de los eventos producidos. Según el tipo de dato a mostrar o editar se utilizan varios tipos de objetos: UITextView para editar campos de texto y UILabel para mostrarlos; UITableView para mostrar la información de forma ordenada por filas y poder seleccionar las celdas, Sliders para la introducción de campos numéricos, UIButtons para la selección de las acciones, etc. La principal forma de organización de la información ha sido mediante tablas, de forma que la información se presenta en cada una de las celdas de las que dispone la tabla. A su vez estas tablas podían ser de contenido estático o dinámico, dependiendo del caso.

4.4. Controladores

Según el tipo de vista se han implementado dos tipos de clases de controlador: clases tipo UITableViewController y clases UIViewController. Cada una de ellas se encarga de manejar, atendiendo al tipo de eventos que se pueden producir, las acciones realizadas por el usuario. A su vez, para cambiar de una vista a otra (y por tanto de un controlador a otro) se ha utilizado la funcionalidad que proveen los storyboards en iOS 5+, más concretamente mediante los llamados 'segues'.

4.5. Notificaciones

A pesar de la idea propuesta en la fase de diseño, para poder mostrar la implementación de las notificaciones y que el profesor no tenga que esperar varias horas como se proponía se ha hecho que aparezca una notificación 20 segundos después de iniciar la aplicación.

```
-(void)scheduleNotification
{
    UILocalNotification *notification = [[UILocalNotification alloc] init];

    // notification is going to fire after 20 seconds
    notification.fireDate = [[NSDate alloc] initWithTimeIntervalSinceNow:20];

    // message to show
    notification.alertBody = @"Remeber to take a look at your uncategorized task!";

    // default sound
    notification.soundName = UILocalNotificationDefaultSoundName;

    // button tittle
    notification.alertAction = @"Go";
    notification.hasAction = YES;

    [[UIApplication sharedApplication] scheduleLocalNotification:notification];
}
```