EMSYS
Modelo de implementación
Versión 14.2

Historia de revisiones

Fecha	Versión	Descripción	Autor
14/11/2016	14.1	Descripción inicial	Juan Manuel San Martín
14/11/2016	14.2	Correcciones	Bruno Amaral

Contenido

1.Introducción

1.1 Propósito

Este documento proporciona una apreciación global y comprensible de la implementación realizada, además de una trazabilidad entre el diseño y la misma. Muestra también las razones por las cuales se tomaron algunas decisiones y menciona los criterios utilizados para implementar de la manera en que se hizo.

1.2 Alcance

Se pretende abarcar toda la implementación realizada y vincularla completamente con el diseño anteriormente realizado.

1.3 Referencias

- Modelo de diseño (v10.1)
- Código de ambos proyectos en liberación beta
- Glosario (v4.2)

2.Frontend

2.1 Patrones

Para la implementación en Frontend se programó utilizando estilos propios de Android, en particular, el patrón MVP(Model View Presenter). El cual es similar al conocido MVC(Model View Controller).

MVP funciona utilizando una vista de android (típicamente una activity o fragment), un modelo (es decir uno o más Dto (Data Transfer Object) y finalmente un presenter, este se compone de una clase, llamada por la vista, la cual se encarga de realizar el funcionamiento de lógica tras la vista.

Por esta razón, el módulo visto en diseño “Lógica Manager” y “ManagerUI” quedan ambas en el paquete “ui” del Frontend.

2.2 Configuraciones

Para las configuraciones se tienen 2 posibilidades, por un lado se utilizaron configuraciones de Gradle, para parámetros que cambian al momento de compilar, y la clase Constants, la cual se encarga de almacenar datos de configuración comunes a varios módulos, típicamente parámetros de comunicación entre estos. Se eligieron estos métodos, debido a que Android no es fácilmente configurable, ya que cambios en archivos usualmente requieren el uso de software especializado, o bien el uso de un ordenador conectado al dispositivo, donde el dispositivo debe darle permisos especiales al ordenador. Se decidió no utilizar archivos almacenados en directorios públicos (como ser los directorios del usuario, descargas, documentos, etc.) del sistema operativo, ya que esto supondría pedir más permisos a la aplicación.

2.3 Database

Se deja para uso a futuro, un paquete database, donde se pueden almacenar datos en una base de datos Sqlite local, especialmente útil para casos de recuperación de la aplicación.

2.4 DaoEvento

El módulo DaoEvento quedó embebido en el paquete managers (donde se aloja el manejador de los eventos) y el paquete logic.core.model, donde se tienen los Dto, y en particular los Dto relacionados a las entidades de negocio.

3.Backend

3.1 Patrones

En la implementación del Backend fue más sencillo seguir la estructura diseñada en etapas anteriores (ver Modelo de diseño), esto es debido a la estructura que se maneja en .Net (y en particular en soluciones C#), donde se divide la solución en proyectos.

Por lo cual cada proyecto contiene uno o más módulos.

3.2 Autenticación y Permisos

Dentro de los casos especiales que vale la pena detallar, se encuentra el proyecto LogicLayer, el cual maneja no sólo el módulo de Lógica especificado en el diseño, sino que también aloja la inteligencia de autenticación, roles y permisos. Este cambio se decidió hacer entrado el proyecto, dado que el Framework (Identity Framework) que se estaba utilizando no cumplía correctamente todos los requisitos que se tenían para el proyecto.

3.3 Configuraciones

También es interesante resaltar cómo se manejaron las configuraciones, las cuales en el diseño se mostraban como el módulo “DaoConfig”, en el caso de la implementación se decidió ubicar las mismas en el archivo web.config dentro de “ServiceLayer”. Esto es así ya que son parámetros de configuración fijos y no variables (idealmente) durante la vida del producto.

4. Trazabilidad

Se indica a continuación la ubicación de los distintos módulos de diseño en los correspondientes de implementación.

4.1 Frontend

	backendCommu-nications	database	logic.model.core	managers	notifications	ui	utils
LogicaManager							
DaoConfig							
ManagerEvento							
ServicioGeolocalizacion							
ManagerUI							
ClienteWS							
ServicioNotificaciones							
DaoEvento							

4.2 Backend

	Service-Layer	DataType-Object	DataAccess-Layer	ObserverData-base	Notifications	LogicLayer
WSProvider						
AutenicacionManager						
PermisosManager						
LogicaController						
DataManager						
DaoConfig						
IservicioExternoConsumer						
IfeNotificaciones						
NotificacionesController						
IdbNotificaciones						