EMSYS
Descripción de la Arquitectura
Versión 5.3

Historia de revisiones

Fecha	Versión	Descripción	Autor
27/08/2016	2.1	Descripción inicial	Juan Manuel San Martín
28/08/2016	2.2	Revisión, cambio de orden de secciones, numeración.	Bruno Amaral
13/09/2016	5.1	Cambio de CU relevantes y agregado de diagramas de diseño	Juan Manuel San Martín
15/09/2016	5.2	Agregado de diagrama de diseño de módulo	Juan Manuel San Martín
18/09/2016	5.3	Arreglos post RTF	Juan Manuel San Martín

CONTENIDO

1.Introducción

1.1. Propósito

Este documento proporciona una apreciación global y comprensible de la arquitectura del sistema usando diferentes puntos de vista para mostrar distintos aspectos del mismo. Intenta capturar y llegar a las decisiones de arquitectura críticas que han sido hechas en el sistema. Y por último pretende justificar éstas últimas de manera correcta, utilizando requerimientos del cliente y técnicas arquitectónicas válidas.

1.2. Alcance

Se pretende abarcar la arquitectura general de la solución que se plantea, junto a esta, se deben justificar las decisiones tomadas.

1.3. Objetivo del sistema

El sistema en cuestión es un gestor de eventos, el objetivo de este, es dar la capacidad de monitorear, actualizar y crear los mismos. En particular se pretende que el sistema sea robusto y de una visión uniforme de los datos en todos los dispositivos donde corra.

1.4.Referencias

- Documento de prototipo 1 (v1.3)
- Documento de prototipo 2 (v3.1)
- Documento de prototipo 3 (v4.1)
- Especificación de requisitos (v4.4)
- Modelo de casos de uso (v4.1)
- Glosario (v4.2)

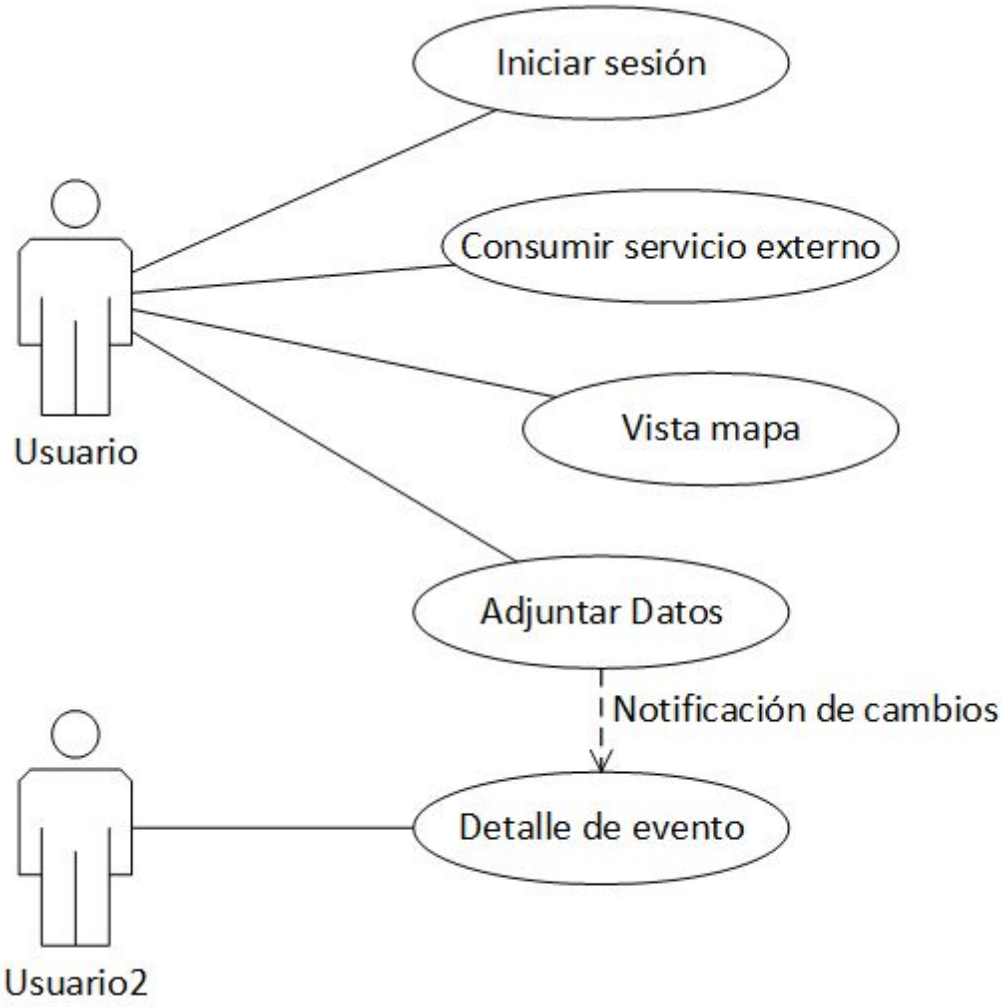
Todos estos documentos se entregarán en el mismo archivo .zip donde se encuentra este.

1.5.Visión general

Se pretende presentar los casos de uso que definen la arquitectura como también la trazabilidad de estos hasta las decisiones arquitectónicas pertinentes.

2.Vista del Modelo de Casos de Uso

2.1.Diagrama de Casos de Uso relevantes a la Arquitectura



2.2. Casos de Uso relevantes a la Arquitectura

En particular, se tomaron estos 5 casos de uso, ya que son los que tienen requerimientos específicos de arquitectura y del diseño y por tanto modelan el sistema.

- Iniciar sesión
- Consumir servicio externo
- Vista mapa
- Adjuntar datos
- Detalle de evento

2.2.1. Iniciar sesión

Este caso de uso es relevante debido a que establece el método de comunicación segura y manteniendo una sesión entre los dispositivos y el backend.

2.2.2. Consumir servicio externo

Consumir un servicio externo, a través del backend determina varias características de este, como por ejemplo la exposición de otro servicio para los dispositivos además de el más claro, el cual es una conexión con un sistema externo utilizando Web Services de tipo SOAP.

2.2.3. Vista mapa

Ver mapas en los dispositivos y ser capaz de colocar capas sobre los mismos determina además de ciertos requisitos técnicos en los mismos, la posibilidad de almacenar capas en backend y que este ofrezca servicios para consumirlos.

2.2.4. Adjuntar datos

En el caso de este caso de uso, se agrega como relevante a la arquitectura por 2 razones, la primera y más clara es la exposición de un servicio del lado del backend, capaz de recibir archivos multimedia (audio, fotos, videos, coordenadas) desde los dispositivos (los cuales además deben ser capaz de consumir este servicio. La segunda razón es que este caso de uso incluye la creación de una notificación a todos los suscriptores de cambios sobre el evento al cual se le adjuntan datos, por lo cual también ayuda en el modelado de la arquitectura al utilizar el canal de notificaciones (requisito de tiempo real) para notificar a los dispositivos de cambios en los eventos.

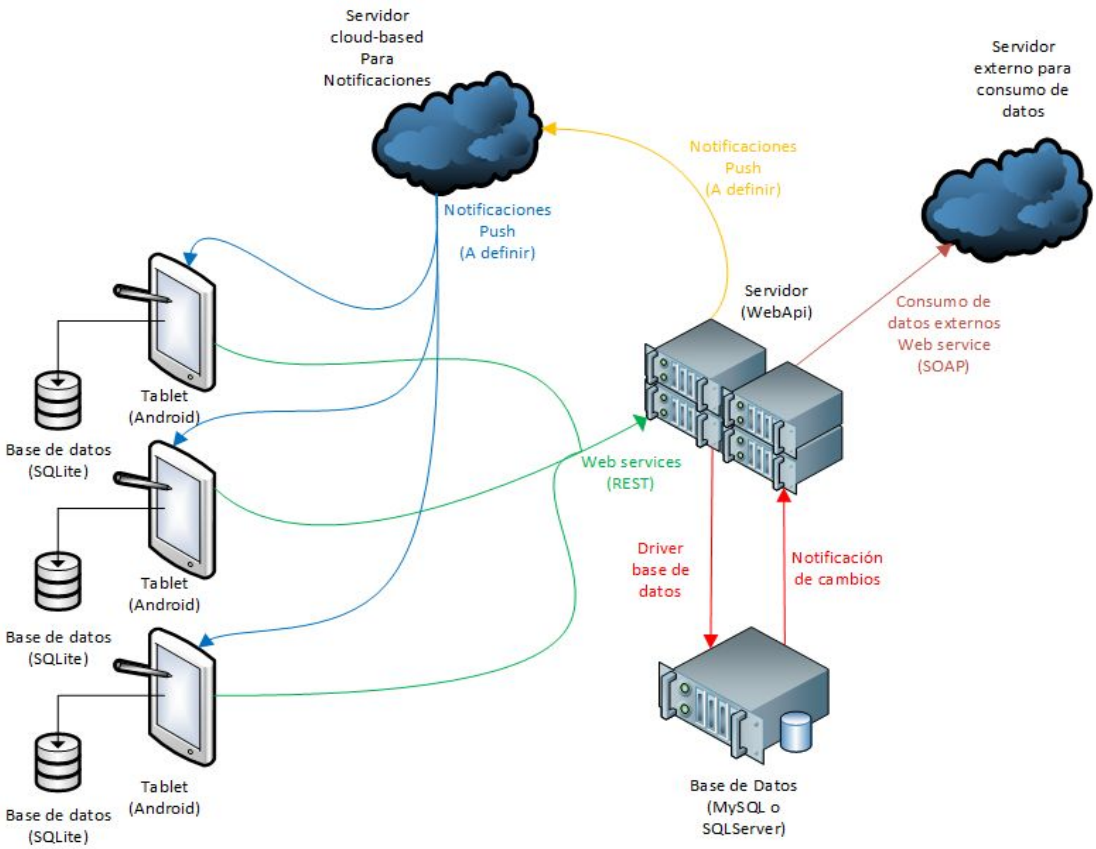
2.2.5. Detalle de evento

Detalle de evento permite agregar todos los elementos faltantes en los casos de uso anteriores, más que nada para determinar todos los conceptos de negocio que aún no se habían visto, como también para modelar la otra fase de las notificaciones de tipo tiempo real, es decir, ver en vivo como cambia la información a medida que se actualice mediante otro dispositivo o cambios en la base de datos. Otro factor no menor que este caso de uso aporta a la arquitectura es el uso de una base de datos local al dispositivo que exhibe el frontend, esta es incorporada para agregar la funcionalidad offline, la cual permite al dispositivo trabajar sin conexión y luego sincronizar sus datos con el servidor.

3.Vista del Modelo de Distribución

3.1.Diagrama de Distribución

De acuerdo a los casos de uso relevantes a la arquitectura se definió la siguiente.



3.2.Nodos

Se pasarán a detallar los nodos relevantes que se visualizan en el Modelo de distribución

3.2.1.Tablet (Android)

Consiste en un dispositivo móvil tipo Tablet, corriendo el sistema operativo Android, sobre el cual se implementará una aplicación cliente, con la cual los usuarios tendrán interacciones. Esta aplicación tiene un contenido lógico mínimo, donde se ofrecerán acciones que impactarán directamente en el Backend (WebApi). Además este dispositivo cuenta con una base de datos SQLite, donde la aplicación podrá persistir datos. Sobre este dispositivo corre el Frontend (detallado más adelante).

3.2.2.Servidor (WebApi)

El servidor consiste en un equipo, corriendo alguna versión de Microsoft Windows e I.I.S., sobre el cual se ejecuta el segundo componente que debemos implementar de tipo WebApi. Esta es la capa lógica del sistema, donde se publican las operaciones que ejecutan las Tablets, se disparan las notificaciones de tipo Push según corresponda, se ejecutan operaciones sobre la base de datos y se consumen los servicios externos que proveen datos. Sobre este servidor correrá el Backend (detallado más adelante).

3.2.3.Base de datos

La base de datos es relacional, aún no se ha definido el motor, dentro de las posibilidades mencionadas por el cliente se manejan MySQL o SQL Server, de este depende la notificación hacia el Servidor de los cambios en la misma. En el prototipo 2 se están investigando métodos para realizar esto, por lo cual, la salida del prototipo probablemente establecerá qué tipo de base de datos se utilizará.

3.2.4.Servidor cloud based para notificaciones

Este componente consiste en un servicio online encargado de repartir las notificaciones a las distintas aplicaciones que corran sobre las tablets, según disponga el servidor. Este servicio se encarga de manejar el nivel de indirección necesario, para que el servidor no tenga la responsabilidad de obtener una conexión saliente para cada dispositivo que corra la aplicación Android.

3.2.5.Servidor externo para consumo de datos

Sobre este servidor el cliente no ha dado grandes especificaciones. Se sabe que este ofrecerá una interfaz de tipo Web Service SOAP, donde el servidor, desarrollado como parte de este proyecto, debe consumir elementos de esta interfaz y luego enviar esta información a las tablets que corran la aplicación Android.

3.2.6.Conexiones

Se describirán brevemente las conexiones que unen varios de estos nodos.

3.2.6.1.Web Services (REST)

Los Web Services de tipo REST son un método de comunicación entre 2 partes, en este caso, se publicará la interfaz de los Web Services en el Servidor y se consumirá desde la tablet que corra la aplicación Android. Estos Web Services ofrecerán toda la información que la aplicación Android necesite de manera de ejecutar todos los casos de uso que comiencen con el actor Usuario.

3.2.6.2.Notificaciones Push

Para esta conexión se ha decidido utilizar Firebase FCM (Firebase Cloud Messaging), consta de mensajes que se envían desde el Servidor hacia las distintas Tablets. Es necesaria para poder contemplar el requisito no funcional de tiempo real. El cual consiste en que las Tablets sean notificadas de cambios en el Servidor sin necesidad de estar realizando operaciones sobre este.

3.2.6.3.Driver de base de datos

Estos drivers se llaman más comúnmente conectores, y consisten en un software encargado de hacer de interfaz entre una base de datos y una aplicación que desea utilizar la misma. En nuestro caso, al no tener establecido que motor de base de datos se utilizará, no se puede saber que driver es necesario aún.

3.2.6.4.Notificación de cambios

Esta conexión tampoco está definida aún, y se creó el prototipo 2 para poder investigar qué opciones existen. Esta conexión es necesaria para que el Servidor pueda ser notificado de cambios externos sobre la base de datos, sin necesidad de estar realizando operaciones sobre esta.

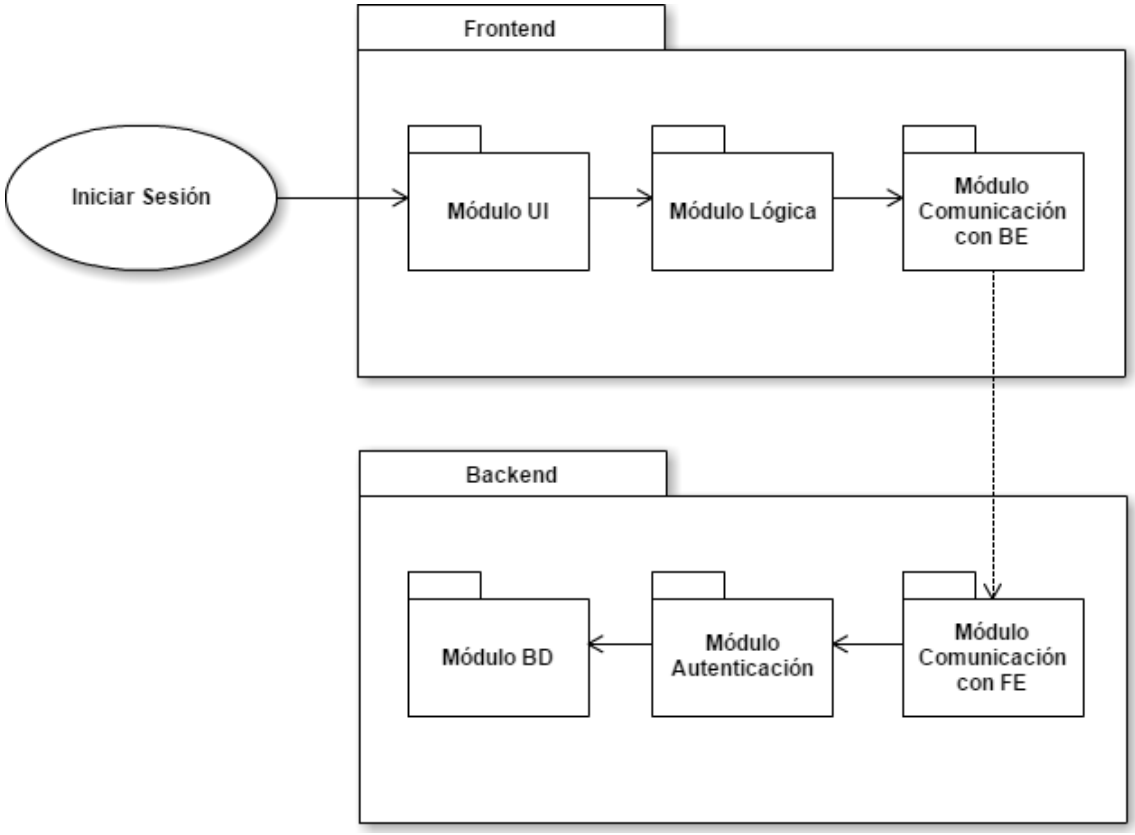
3.2.6.5.Consumo de datos externos (Web Services SOAP)

Esta conexión consiste en el consumo de Web Services de tipo SOAP, estos son un tipo de Web Services, los cuales existen para comunicar 2 entidades, y funcionan publicando métodos en una de ellas, mientras que la otra debe consumir estos últimos. Sobre estos el cliente sólo ha especificado que son de tipo SOAP y que probablemente la firma tenga un formato sencillo.

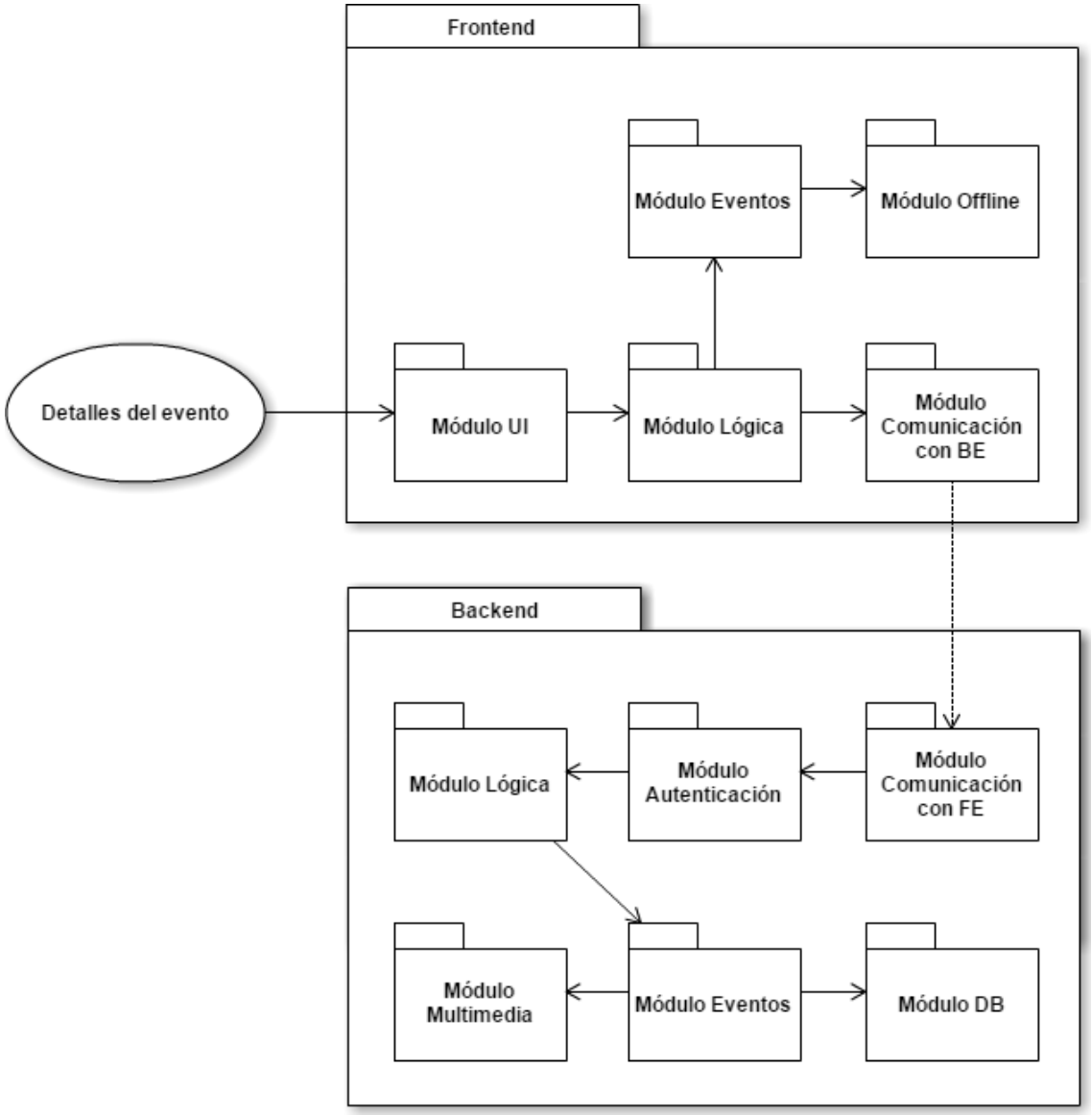
4. Distribución en Módulos

4.1. Trazabilidad desde el Modelo de Casos de Uso al Modelo de Diseño

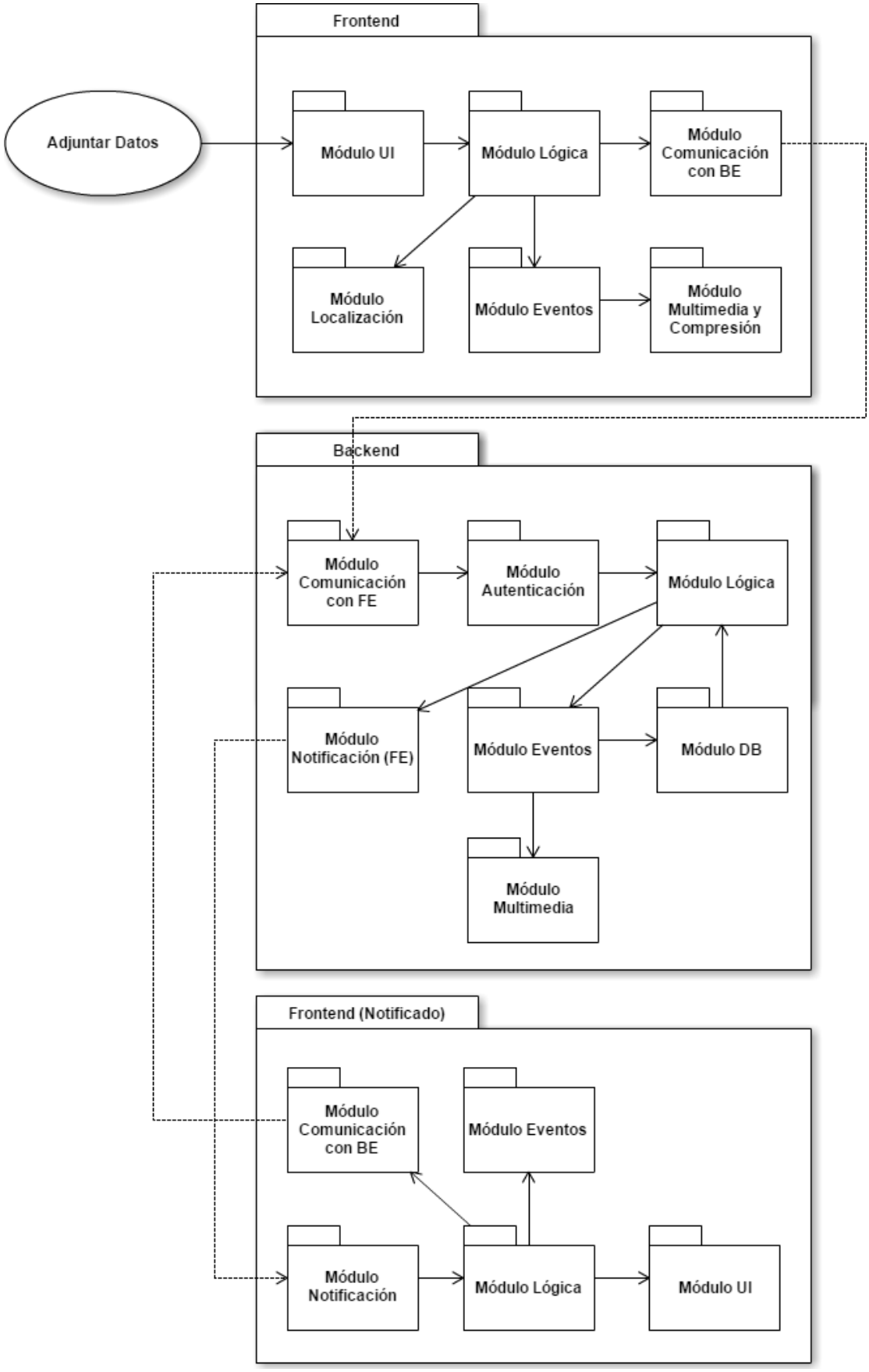
4.1.1. Iniciar sesión



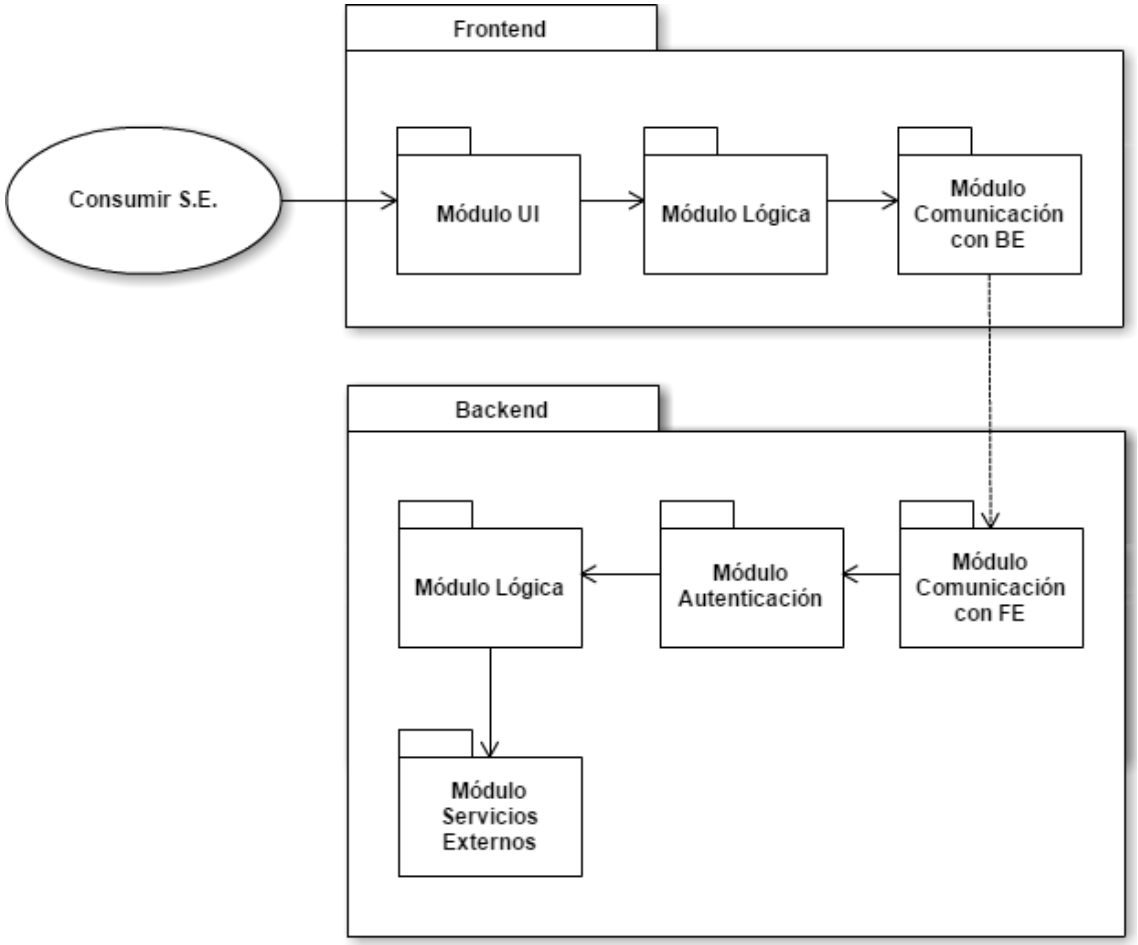
4.1.2. Detalles del evento



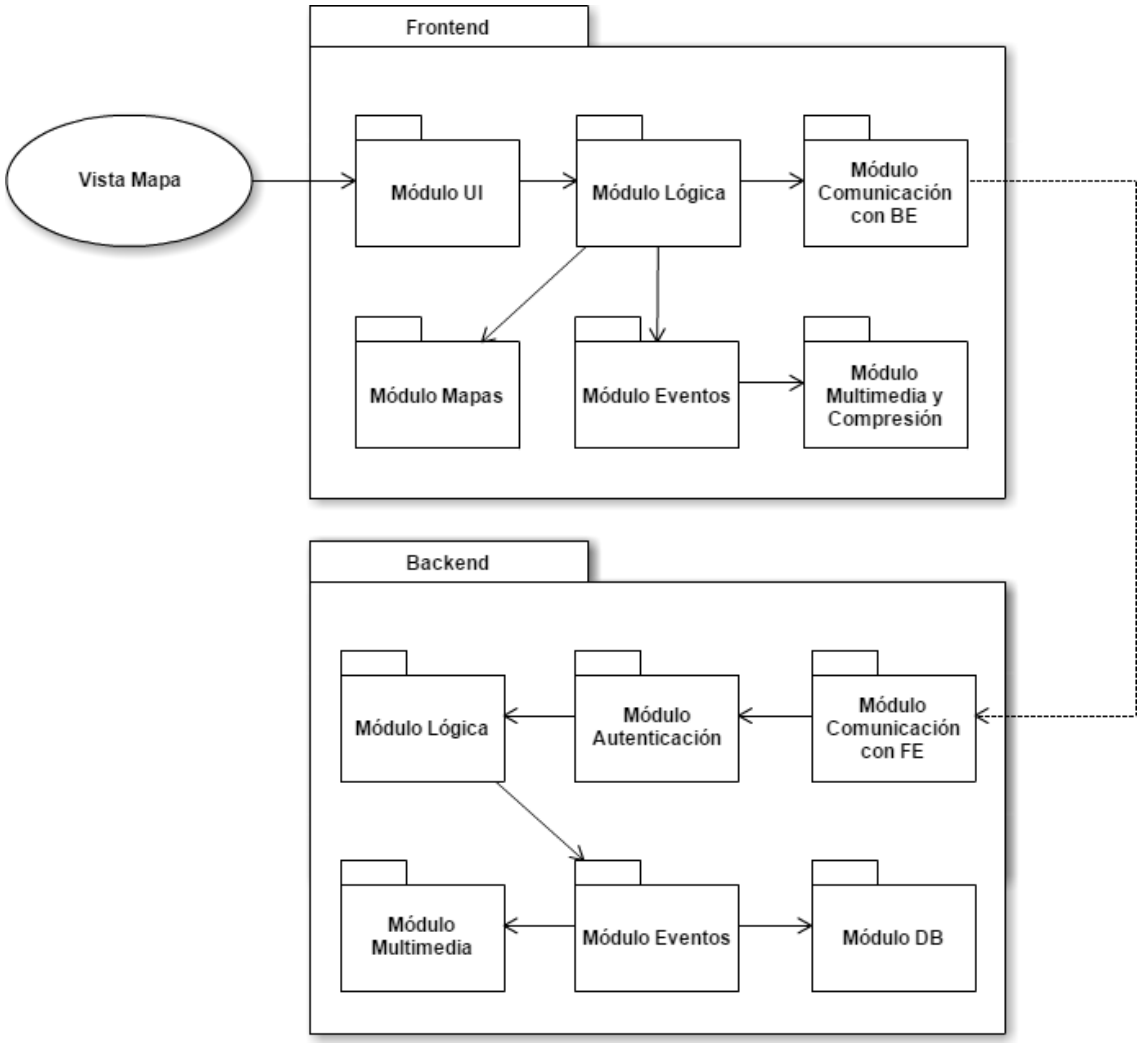
4.1.3. Adjuntar Datos



4.1.4. Consumir servicio externo

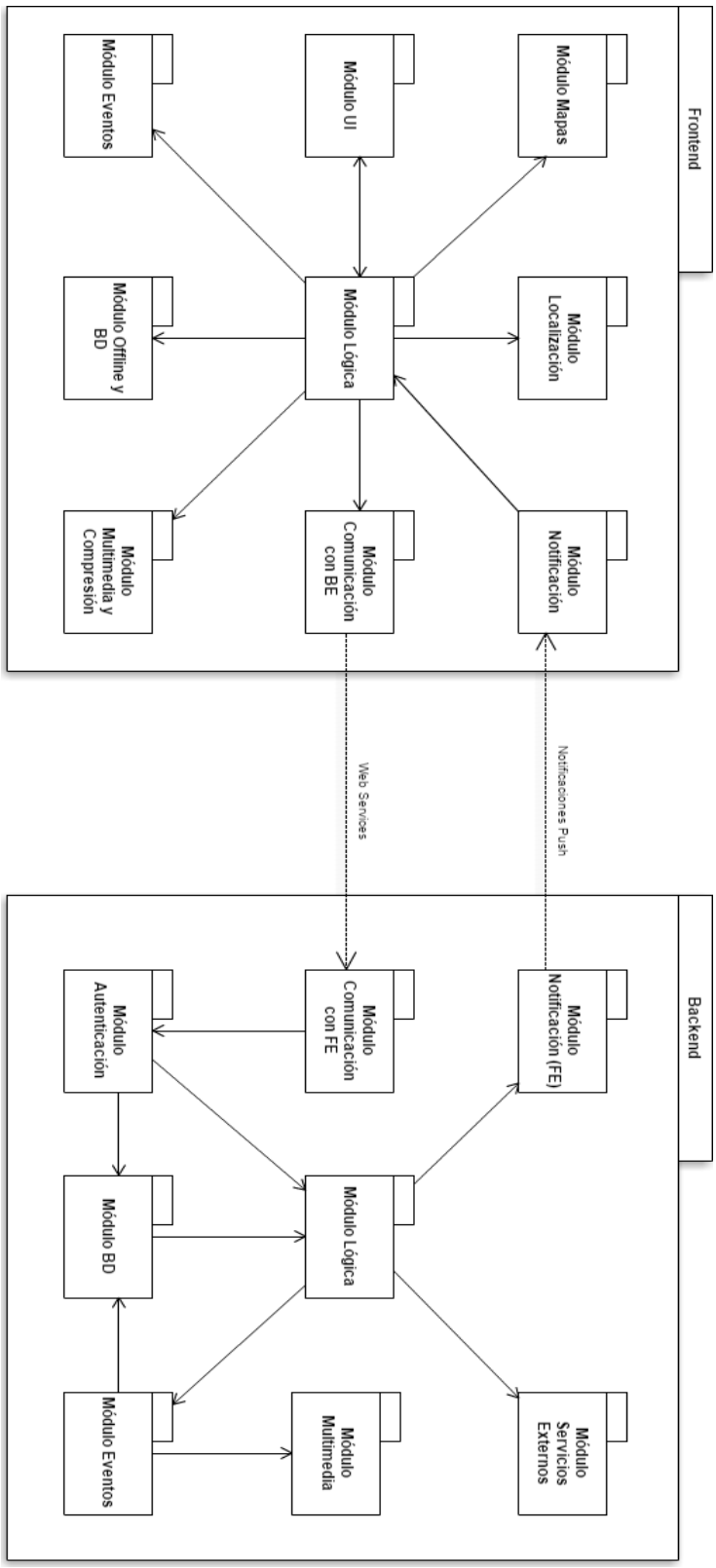


4.1.5. Vista mapa

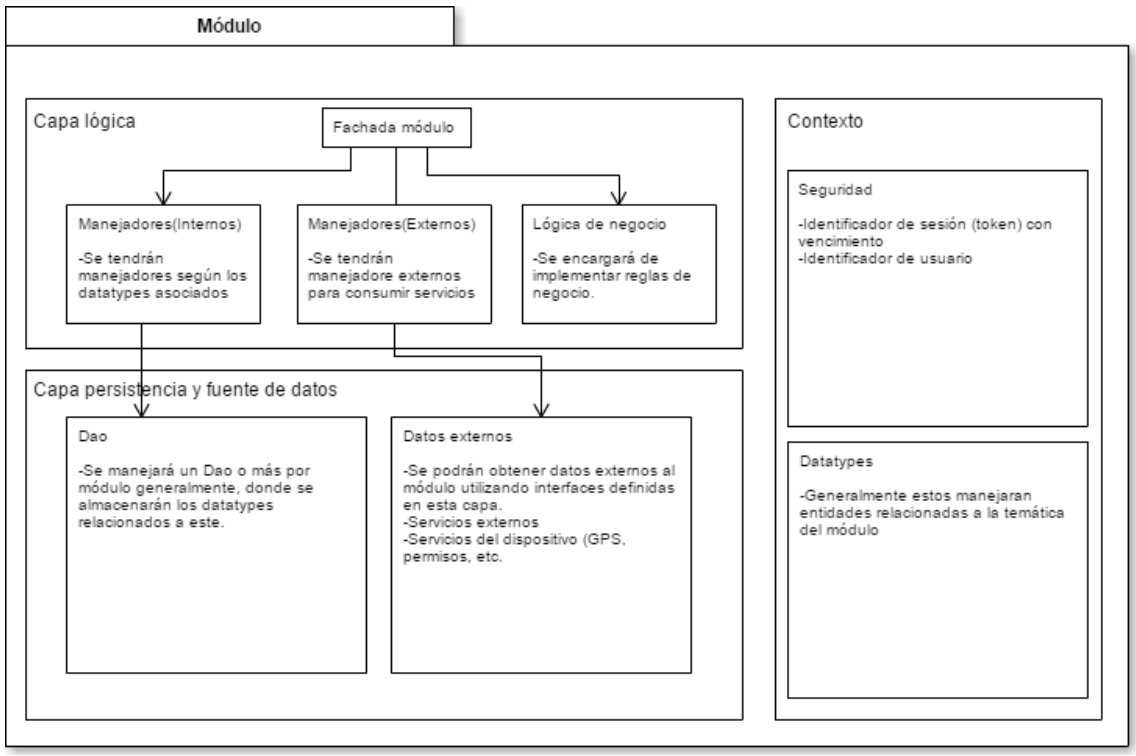


4.2. Vista del Modelo de Diseño

4.2.1. Descomposición en módulos



4.2.1. Vista interna de un módulo



Este esquema modela un módulo genérico, donde un módulo no denota una "carpeta" dentro del código donde se almacenan todas los elementos relacionados, sino que significa que existe un conjunto de elementos donde existe la relación mostrada en el diagrama.

- Contexto

El contexto incluye la información que está disponible en toda la secuencia de pasos que siga un caso de uso dentro del mismo módulo. Se incluye información del usuario, fundamental para determinar el funcionamiento de algunos casos de uso.

- Capa lógica

Dentro de la capa lógica se aloja todas las reglas de negocio y algoritmos relacionados con la temática del módulo. Existen manejadores internos que tienen el poder de acceder a datos locales, o manejadores externos, que utilizan otros medios para obtener datos, mientras que por otra parte existe un elemento Lógica de negocio, que se encarga de aplicar las reglas de negocio necesarias para cumplir de forma semánticamente correcta con los casos de uso.

- Capa persistencia y fuente de datos

La capa persistencia y fuente de datos se encarga de manejar las entradas de datos para su posterior manejo. En esta capa se encuentran, tanto elementos capaces de persistir localmente datos, como otros que son capaces de obtener datos de fuentes externas, utilizando en nuestro caso particular, Web Services de tipo SOAP y REST.