SICC

Descripción de la Arquitectura Versión 1.2

Historia de revisiones

Fecha	Versión	Descripción	Autor
26/08/2016	1.0	Creación del documento	Andrés Bello
27/08/2016	1.1	Incorporación de subsistemas	Andrés Bello
28/08/2016	1.2	Revision de SQA	Juan Nogueira

Contenido

1. Introducción

En una primera instancia se mostrará el diagrama de despliegues, el cual ilustra cómo se distribuirá el sistema en los diferentes componentes físicos. Además se muestra un diagrama de paquetes, indicando como deberá estar constituido cada componente del sistema.

2. Propósito

Este documento proporciona una apreciación global y comprensible de la arquitectura del sistema. Intenta capturar y llegar a las decisiones de arquitectura críticas que han sido hechas en el sistema.

3. Alcance

En esté documento se pretende indicar como estará formada la arquitectura global del sistema, es decir, como esta distribuida físicamente y como esta formado cada nodo. Además en futuras iteraciones se mostrará como los casos de usos relevantes para la arquitectura interactúan con los componentes de la misma.

4. Definiciones, siglas y abreviaturas.

MVC – Model View Controller

MVP – Model View Presentator

5. Referencias

No aplica

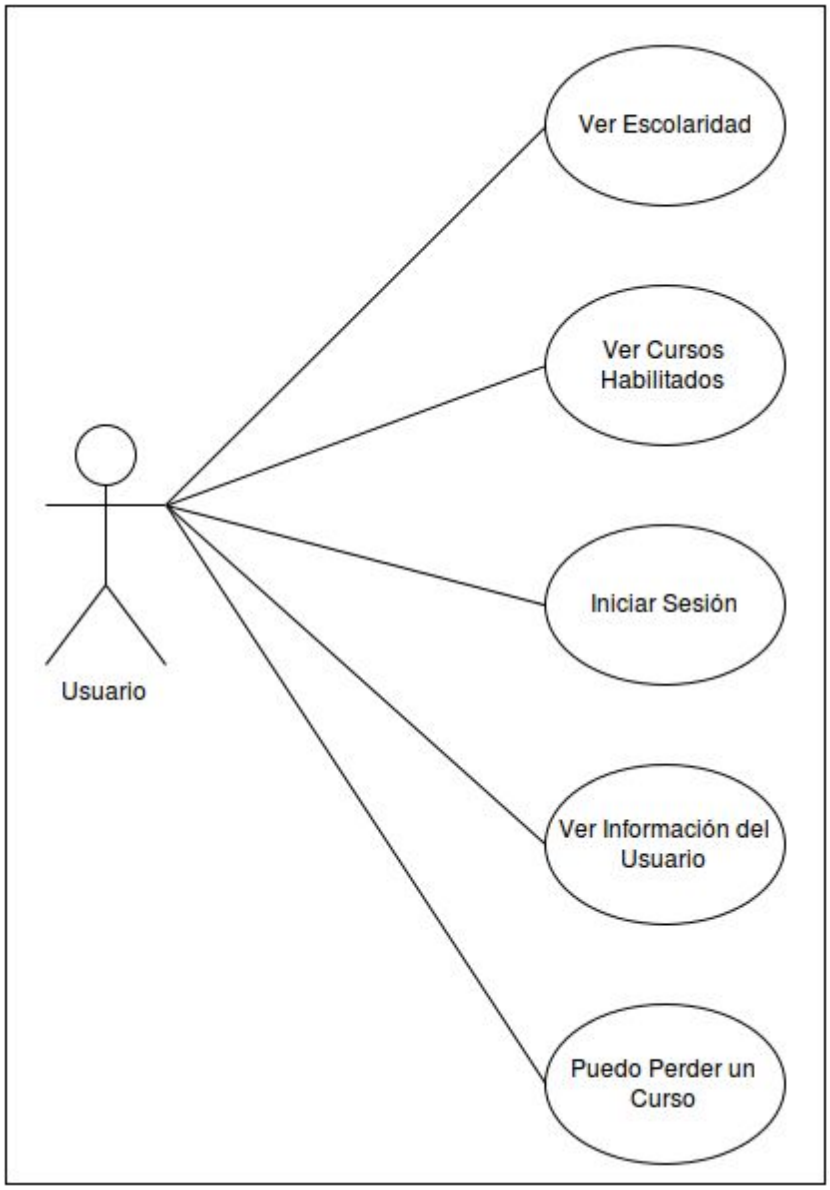
6. Visión general

No aplica

7. Vista del Modelo de Casos de Uso

7.1 Diagrama de Casos de Uso relevantes a la Arquitectura

Se incluye un diagrama de casos de uso, con aquellos que son relevantes en la arquitectura.



La especificación de cada caso de uso se detalla en el Modelo de Diseño

8. Trazabilidad desde el Modelo de Casos de Uso al

9. Modelo de Diseño

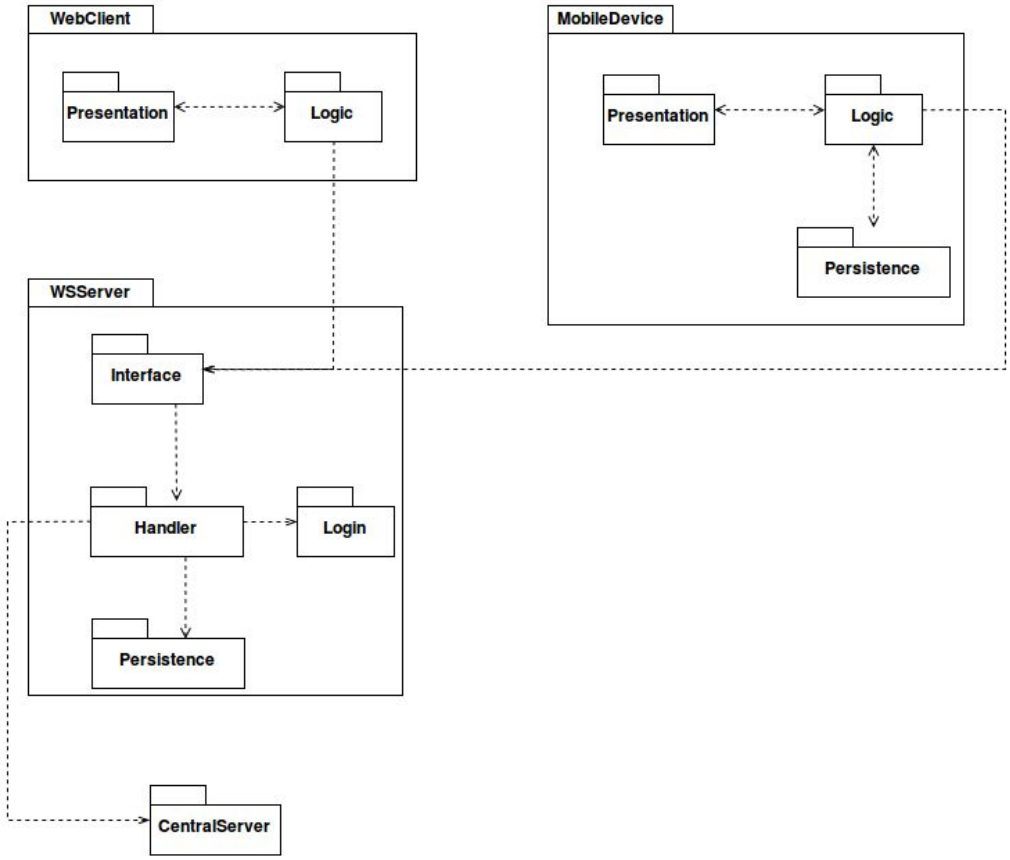
No aplica para esta iteración

9.1. Caso de Uso relevante a la Arquitectura 1

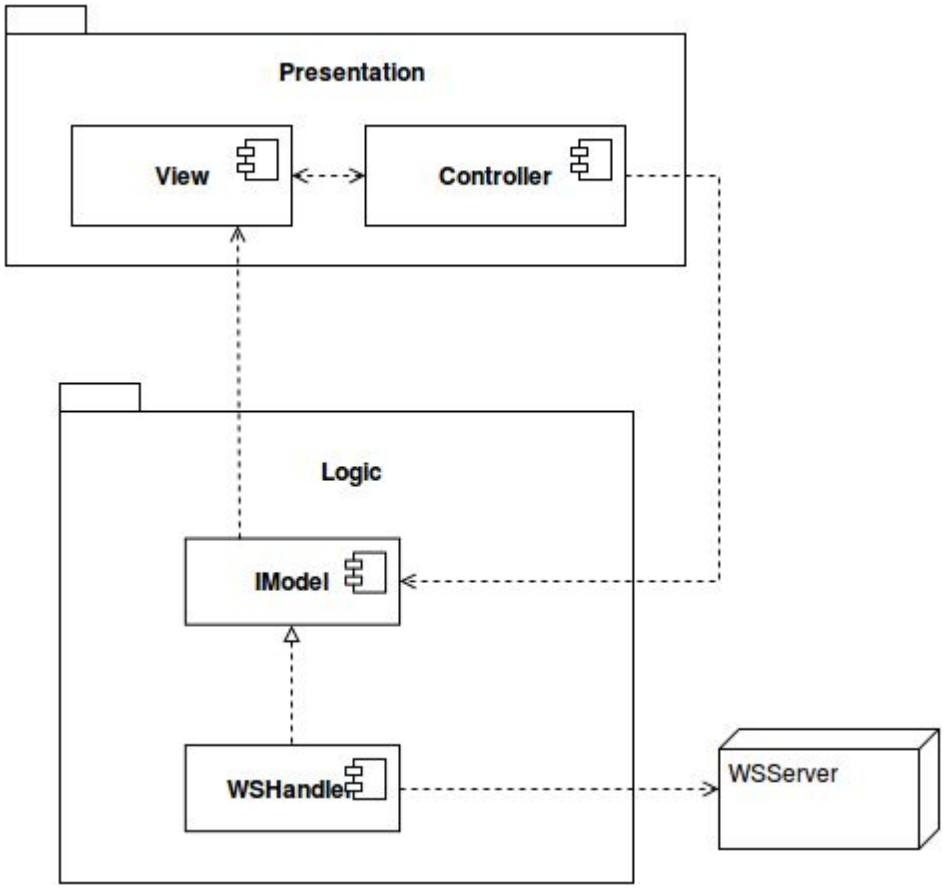
No aplica para esta iteración

10. Vista del Modelo de Diseño

10.1. Descomposición en Subsistemas



WebClient



Se utilizó el patrón de arquitectura MVC (Model View Controller) para diseñar el cliente web. Se distinguen las siguientes componentes:

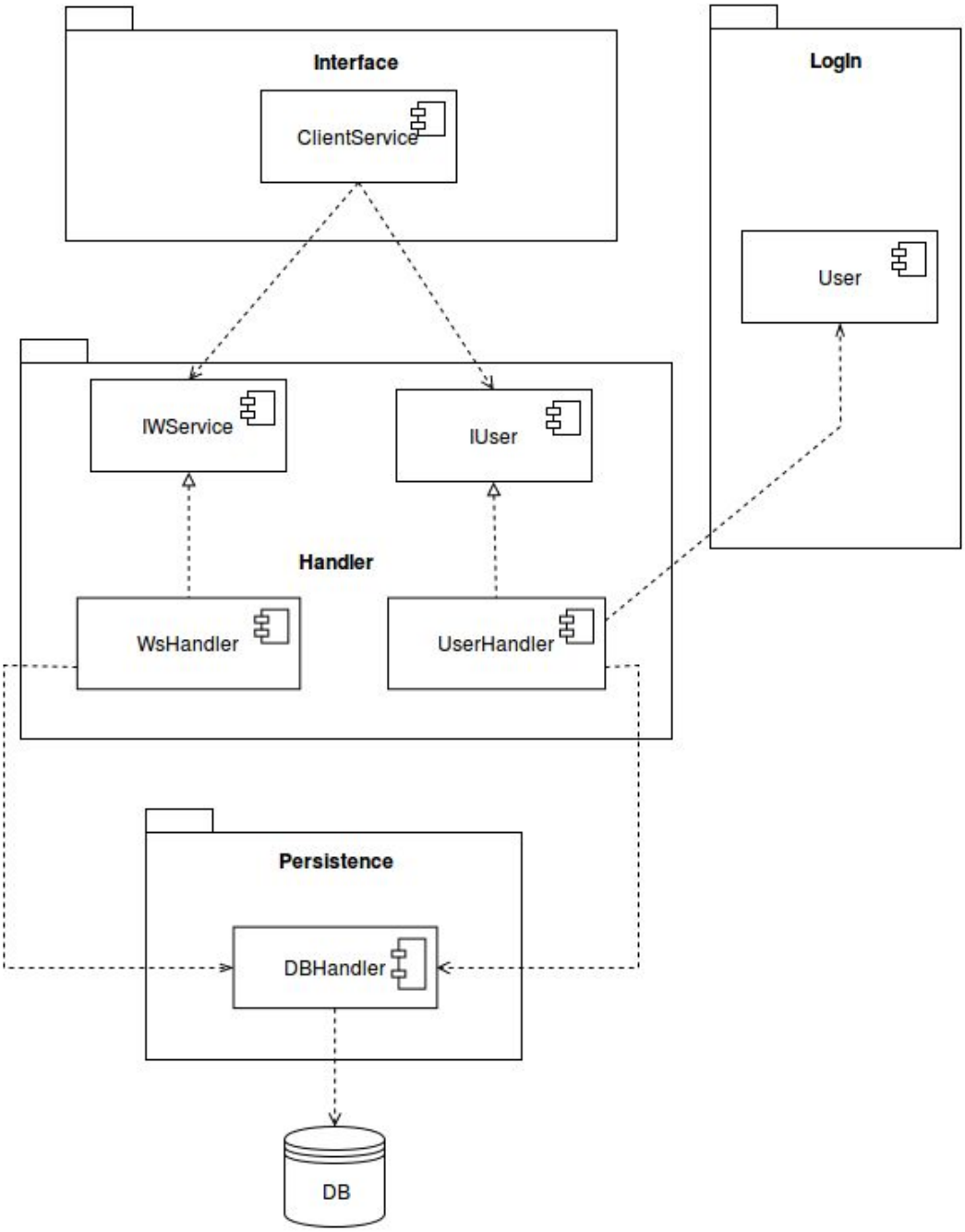
1. Presentation
   Contiene dos modulos, el modulo View, que implementa las vistas UI, y es la interfaz con el usuario. El modulo Controller captura las acciones provocados por los usuarios e invoca a las operaciones de IModel para resolver las peticiones. Una vez que obtuvo el resultado lo envía a la vista, eventualmente provocando un cambio.
2. Logic
   Contiene el modulo de modelo, diseñado por una interfaz y un manejador. El modulo Controller interactua con IModel. WSHandler implementa IModel, siendo el responsable de resolver los pedidos y devolver los resultados a Controller o a View directamente. A su ves WSHandler interactua con el servidor central para resolver las operaciones.

MobileDevice

El diseño para el cliente móvil es bastante parecido al cliente web. Salvo que se utilizó el patrón MVP (Model View Presentator) y una base de datos que utilizan los sistemas operativos Android como forma de cache. Por lo tanto se distinguen los siguientes componentes:

1. Presentation
   Contiene la vista y el presentador, ViewModel y Handler respectivamente. ViewModel implementa la UI y es con quien el usuario interactuá, Handler captura las acciones que este provoca y las pasa al modelo para que este las resuelva. Su rol es de intermediario entre la UI y la lógica de negocio.
2. Logic
   Implementa el modelo. El cual esta constituido por una interfaz (Ilogic) y un manejador que implementa la interfaz (WSHandler). Este último es el encargado de resolver la lógica de negocio y comunicarse con el servidor central cuando sea necesario. Ademas es quien conoce al manejador de persistencia.
3. Persistence
   Los dispositivos Android cuentan con una base SQL Lite que por lo general se utiliza como cache. Por lo que se diseño un manejador (AndroidDBHandler) que sera el responsable de realizar todas las operaciones de base de datos

WSCenter



El servidor central es un módulo Cliente Servidor grueso, su rol de cliente lo ejerce cuando debe consumir servicios del servidor central y su rol de servidor cuando expone servicios para el cliente web y el cliente móvil.

Es el responsable de resolver toda la lógica de negocio, por ejemplo, procesar los datos obtenidos de los servicios del servidor central, realizar la autenticación de usuario, etc.

Se diseñó utilizando cuatro capas:

1. Interface
   Esta capa es la encargada de exponer los servicios a los clientes finos.

2. Handler
   Esta capa implementa la lógica de negocio, esta compuesta por dos interfaces, con las cuales se comunican la interfaz de servicios y sus correspondientes manejadores.
3. Persistence
   Capa encargada de manipular la base de datos
4. LogIn
   Esta capa está dedicada especialmente a la funcionalidad de LogIn, el cliente manifestó que es posible que la forma de iniciar sesión puede variar, por lo que se separó en otra capa para que luego sea fácil reemplazarla.

11. Diseño de Clases

Se especifica en el documento de Modelo de Diseño

11.1. Interfases

A la espera de que el cliente nos libere la API para ver los servicios que ofrece y las interfaces que debemos crear.

12. Vista del Modelo de Implementación

No aplica a esta iteración

13. Vista del Modelo de Distribución

13.1. Diagrama de Distribución

Es esta sección se presenta un diagrama que ilustre las distintas componentes físicas, las conexiones y como esta formado cada componente es esta etapa inicial.

13.2. Nodos

A continuación se detallan los nodos que se creen necesarios en una primera instancia.

MobileDevice

Cliente fino cuyo objetivo es que los usuarios interactúen con el sistema mediante teléfonos móviles con sistema operativo Android.

WebClient

Los usuarios tienen la posibilidad de acceder al sistema mediante un cliente fino web, el cual es de utilidad para abarcar a los usuarios que no cuentes con dispositivos Android.

WSServer

Es un componente cliente servidor. Puesto que tiene como objetivo ser el intermediario entre los clientes, tanto móvil como web, y el servidor provisto por el cliente, el cual contiene la API que brindará las principales funcionalidad y la base de datos).

CentralServer

El servidor central es proporcionado por el cliente, en el se encuentra la base de datos principal con la información de los estudiantes y además brinda una API para poder consumir los servicios que brinda la aplicación que hay desarrollada hoy en día.

13.3. Conexiones

Para todas las conexiones se deberá utilizar protocolos de comunicación, por ejemplo API Rest, utilizando HTTPS.