UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

DESDE LA WEB A LOS MÓVILES: LLEVANDO U-CURSOS A NUESTROS BOLSILLOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

ALFREDO JAIME CÁDIZ RODRÍGUEZ

PROFESOR GUÍA:
JAVIER BUSTOS JIMÉNEZ

MIEMBROS DE LA COMISIÓN:
JOHAN FABRY
PATRICIO INOSTROZA FAJARDIN

SANTIAGO DE CHILE
ENERO 2014

Resumen

U-Cursos es una herramienta para dar apoyo y mejorar la docencia presencial. Es desarrollada y mantenida por ADI (Área de Desarrollo de Infotecnologías) de la Escuela de Ingeniería de la Universidad de Chile. Actualmente posee más de 20 servicios y más de 30.000 usuarios activos entre docentes, funcionarios y alumnos de la institución. En sus más de 10 años de existencia, la plataforma se ha convertido en el principal medio de comunicación de la Universidad. Esto porque ofrece servicios desde foros y novedades hasta la publicación de notas parciales y actas finales.

A pesar de ser una herramienta que se ha vuelto indispensable para alumnos y docentes de la Universidad de Chile, presenta algunas limitaciones para usuarios que necesitan aprovechar de mejor manera el sistema. Actualmente no se explota la gran cantidad de teléfonos inteligentes y conexiones a internet móvil presentes en el mercado chileno. Tampoco ofrece facilidades para desarrolladores externos que deseen complementar los servicios de la plataforma, obligándolos a preocuparse de procesar la información desde las vistas HTML antes de poder generar valor a partir de los datos.

En esta memoria proponemos el diseño e implementación de un cliente móvil de U-Cursos que permite el acceso eficiente a la plataforma desde tales dispositivos. Esto consiste en un mejor despliegue de los contenidos y de organización de la información, adaptándose a sus pantallas de menor tamaño. Además aprovechamos las características inherentes de tales dispositivos, como su conexión permanente a internet, para mejorar la comunicación con los usuarios. Para esto, primero analizamos la plataforma mediante el procesamiento de los datos de sus servidores y a través encuestas a sus usuarios. Luego, basados en la información recopilada, proponemos una arquitectura para dar soporte a un servicio de APIs de contenidos de U-Cursos y para permitir el despacho de notificaciones a dispositivos móviles.

Dado lo anterior, proveemos un cliente móvil de U-Cursos implementado para la plataforma iOS. Este cliente ofrece acceso a contenido adaptado para teléfonos inteligentes y recibe notificaciones de eventos relevantes para el usuario generados desde la plataforma. Adicionalmente definimos las APIs necesarias para facilitar el acceso a los recursos del sistema desde la aplicación móvil.

Agradecimientos

En primer lugar, gracias a mis padres por darme soporte todos estos años, por confiar en mis decisiones y cambios de rumbo del curso *normal* de los estudios. También a mis primos que son fuente de inspiración y orgullo por sus logros en sus respectivos áreas. A mi hermana y tía Janel por estar siempre disponible a apoyarme. A mis abuelos por su cariño incondicional y en general a toda mi familia.

Cabe también destacar el gran trabajo de los profesores del colegio en darme una formación como estudiante y persona que me han permitido lograr grandes cosas, en especial a Patricia Gamboa por sus clases magistrales de cálculo y por la motivación por el estudio de las ciencias.

Tampoco puedo dejar de mencionar a la gran fuente de motivación en época de universidad: Jose Miguel Piquer quien con sus clases era imposible perder ánimos en continuar la senda de las ciencias de la computación.

Gracias al equipo docente y organizador del programa *EMOOSE* el que además de abrirme la mente aún más respecto a mi formación, me permitió conocer personas maravillosas tanto en Francia como en Bélgica. Entre las que no puedo dejar de nombrar a Boris, Coen, Elisa, Jorge y Wolfgang.

A Javier Bustos, quién fue profesor en la U, luego compañero de trabajo, ahora jefe en NIC Chile Research Labs y profesor guía de esta memoria, por confiar en mis capacidades.

Agradecimientos al equipo de ADI por permitirme realizar la presente memoria y ofrecerme todo el soporte necesario para que fuese exitosa.

También a mis queridos y cercanos que a pesar de todo, han estado en las buenas y malas: Ariel, Dusan, Gonzalo, Osvaldo, Rodrigo y Tomás. Y además a mis amigos del colegio con los que he pasado tantas etapas: Andrea, Daniel, Francisco, Ignacio, Jorge, Marcelo y Rosita.

Finalmente quisiera agradecer a los proyectos Corfo Innova 12IDL1-15665 y 12IDL-16017 que financiaron parcialmente la realización de esta memoria.

Tabla de contenido

Índice de tablas

Índice de figuras

Introducción

El software para computadores personales ha evolucionado a grandes pasos en las últimas décadas. Inicialmente las aplicaciones eran concebidas como instancias que se ejecutaban de manera local, aisladas en cada computador, y las comunicaciones de redes eran casi exclusivamente para programas especializados como los clientes de mail o de transferencia de archivos.

Con la popularización de internet durante la década de los noventa, los programas se beneficiaron de oportunidades que no eran posible antes, como actualizaciones en línea y permitir la transmisión de información entre pares. Además se ven las primeras nociones de las posibilidades de la web, como el despliegue de información utilizando el estándar HTML (HyperText Markup Language) que podía ser desplegado por cualquier navegador. En sus inicios la web era principalmente estática y la industria aún dependía mucho de la instalación de cada una de sus aplicaciones en cada computador. Desde el año 2000, la internet ha tenido un crecimiento en usuarios sin comparación, llegado al año 2013 con un estimado de 2.2 billones de usuarios conectados y las expectativas apuntan que esta tendencia se mantendrá [6, 14]. Esto ha sido impulsado principalmente por la llamada *WEB 2.0* [11], que consiste en la nueva forma de ver internet como una plataforma de servicios en la que los usuarios pueden interactuar tanto con proveedores de contenidos como con otros usuarios. Tecnologías estandarizadas como HTML, Cascade Style Sheets (CSS) y JavaScript han permitido que las aplicaciones web actuales, a pesar de ser mucho más complejas en comportamiento y contenido, puedan ser presentadas en cualquier navegador que las soporte.

La proliferación de servicios basados en la web y la mejora en la calidad de la conectividad inalámbrica ayudó a que la computación móvil tomara fuerza y se hiciera tremendamente popular. Sistemas operativos móviles como Symbian de Nokia y Blackberry OS de RIM aprovecharon la capacidad de las redes existentes para dar un valor agregado a sus servicios. En particular Blackberry se posicionó por su eficiente manejo del correo electrónico. Desde el año 2007 con el lanzamiento del primer iPhone, la ya existente industria de los teléfonos inteligentes tomó mucho más fuerza cambiando el foco de atención de los computadores de escritorio hacia los dispositivos móviles. En 2011 se alcanzó un punto de inflexión, donde los teléfonos inteligentes sobrepasaron en ventas a los computadores de escritorio [2]. Los dispositivos móviles modernos lograron posicionarse como poderosas herramientas para sus usuarios. Esto es porque integran sensores (movimiento, luz, acercamiento), geoposicionamiento y servicios como las notificaciones push y sincronización con servidores *en la nube*, que junto a una conexión permanente a internet proveen una experiencia personalizada y aumentada que un computador de escritorio o incluso portátiles no pueden equiparar.

Los principales actores de la actual revolución móvil son Google con su sistema operativo Android y Apple con iOS . Estos sistemas operativos además de ofrecer prestaciones acordes a las capacidades del hardware actual, proveen a los desarrolladores terciarios de Kits de Desarrollo de Software (SDK por sus siglas en inglés) robustos, incluyendo lenguajes de programación, gran cantidad de librerías y una curva de aprendizaje apropiada para la mayoría de los programadores [9].

La masividad actual de los dispositivos móviles y SDKs apropiados para los desarrolladores ha llevado a las aplicaciones móviles de ser una curiosidad a ser una necesidad para sus usuarios. Esto se refleja en los datos de Google y Apple, quienes a fines de 2012 anunciaron que sus respectivas tiendas de aplicaciones móviles poseían aproximadamente 700.000 aplicaciones activas[1]. Adicionalmente la calidad de las aplicaciones se va tornando más importante para mantener a los usuarios leales a la ellas[2]. Para los proveedores de servicios en línea, el crecimiento del mercado móvil impone el desafío de asegurarse que sus sitios puedan ser accedidos por estos dispositivos.

Dada la importancia y gran adopción de los servicios basados en internet, en el año 2002 la Facultad de Ciencias Físicas y Matemáticas (FCFM) de la Universidad de Chile constituyó el Área de Desarrollo de Infotecnologías (ADI). Este grupo esta a cargo de automatizar los procesos académicos de la facultad utilizando tecnologías de la información a través de servicios en línea. Uno de sus servicios más exitosos es U-Cursos [4], una plataforma de apoyo a la docencia que tiene como objetivo mejorar la gestión y coordinación de los cursos impartidos por la Universidad de Chile. U-Cursos es un portal que ofrece al usuario el acceso a todos los cursos en los que esté involucrado, tanto alumno o como docente. Posee instancias de publicación de contenidos, intercambio de opiniones y además de publicación de evaluaciones. La última gran actualización del servicio fue incluir comunidades [3], las cuales permiten acercar a los usuarios con gustos afines en grupos que trascienden los semestres académicos.

Actualmente este servicio tiene un único punto de acceso, en `http://www.u-cursos.cl`, que corresponde a un portal ofreciendo contenido en HTML que puede ser desplegado desde cualquier navegador moderno. Pese a ello, dada la gran cantidad de elementos desplegados actualmente, el acceso a los servicios por medio de dispositivos de pantalla pequeña es complicado y lento ya que se debe interactuar con una disposición de los elementos pantalla poco óptima. Esto se refleja en los bajos índices de acceso al servicio a través de dispositivos móviles. También, dado que no hay acceso fácil a los datos del servicio, las oportunidades de generación de valor agregado por parte de personas externas a ADI son casi nulas, ya que obliga al desarrollador a procesar la información desplegada en las vistas HTML con los riesgos de que cualquier cambio de formato afecte a su aplicación. Además, se pierden oportunidades de utilizar tecnologías modernas para notificar a los usuarios de actualizaciones relevantes en sus cursos.

En la presente memoria propondremos la implementación de un cliente móvil en el sistema operativo iOS de Apple y la arquitectura necesaria para ello. Para lograr esto analizamos la situación actual de U-Cursos respecto al acceso desde móviles y consultamos a los usuarios

[1] http://www.businessweek.com/news/2012-10-29/google-says-700-000- applications -available-for-android-devices

[2] http://www.localytics.com/blog/2012/app-user-loyalty-increasing-ios-beats-android

sobre sus expectativas de un cliente móvil. Luego recopilamos los requisitos desde ADI y de los usuarios de U-Cursos para proponer una lista de requerimientos de un cliente móvil. Además proponemos implementar una API abierta del sistema que permita el acceso de terceras personas a través de puntos de accesos bien documentados y seguros. Para desarrollar lo anterior, definimos una arquitectura del servidor para dar soporte a las funcionalidades deseadas para el cliente móvil. Finalmente implementamos la aplicación para iOS y los componentes necesarios para la autenticación y el acceso a los datos.

El resto del documento se estructura de la siguiente manera: primero presentamos la motivación, objetivos y metodología en el Capítulo 1, a continuación revisamos el estado del arte en el Capítulo 2. En el Capítulo 3 analizamos los datos de los servidores de U-Cursos y detallamos la encuesta a los usuarios y los resultados obtenidos. A continuación, en el Capítulo 4 revisamos los requisitos de cliente móvil y las APIs de U-Cursos para proponer un diseño de software y arquitectura del servidor, para luego en el Capítulo 5 detallamos cómo se han implementado los distintos componentes del sistema. Los resultados del trabajo realizado se resumen en el Capítulo 6 y finalmente concluimos en el Capítulo 7.

Capítulo 1

Motivación, objetivos y metodología

En este capítulo indicamos la motivación de la presente memoria y el valor que entrega su implementación. Dado esto, definiremos nuestros objetivos generales y específicos. Finalmente planteamos la metodología utilizada para el desarrollo del trabajo.

1.1. Motivación

U-Cursos es sin duda uno de los servicios más utilizados entre los ofrecidos por ADI, posee una gran población activa de alumnos y docentes que coordinan cursos y actividades extra-curriculares a través de él. Lamentablemente la plataforma se ha mantenido en el ámbito de los computadores de escritorio dado que el acceso desde teléfonos inteligentes no es óptimo ni eficiente. Esto ocurre porque la plataforma que no adapta su contenido para desplegarse en pantallas pequeñas generando dificultades en su uso. Además se pierden oportunidades de mejorar la interacción con los usuarios al no aprovechar tecnologías actuales, como las notificaciones push. Adicionalmente, por su gran actividad diaria la plataforma genera gran cantidad de información que actualmente sólo es usada por los servicios provistos por sus desarrolladores. A pesar de ello, varios proyectos han surgido a partir del procesamiento de la información desplegada en por las vistas HTML de U-Cursos, lo que demuestra que existe interés en generar servicios nuevos a partir de los datos existentes.

Con el trabajo de la presente memoria buscamos satisfacer la necesidad de un cliente móvil para U-Cursos, esto facilitará la interacción de la plataforma con los usuarios al ser más eficiente al desplegar información y al recibir notificaciones de actualizaciones de contenidos.

Además desarrollaremos un primer conjunto de APIs de acceso a los contenidos de la plataforma para ser usados por desarrolladores. Tales APIs permitirán que programadores externos a ADI implementen aplicaciones que complementen las funcionalidades de U-Cursos con menor incertidumbre sobre la forma de obtener los datos del sistema.

1.2. Objetivos

En esta sección describiremos los objetivos generales y específicos para la presente memoria. Los objetivos específicos nos permitirán explicitar los hitos que deberán cumplirse para lograr completar el objetivo general.

1.2.1. Objetivo general

Mejorar la interacción entre los usuarios de U-Cursos ofrecer acceso desde teléfonos inteligentes mediante la implementación de una aplicación nativa y la definición de APIs de acceso a los datos.

1.2.2. Objetivos específicos

Estudiar situación actual de los servicios de U-Cursos: Tal estudio convergerá en un reporte del uso histórico de la plataforma y un análisis de los servicios más relevantes de su versión web. Esto incluirá métricas generales como la cantidad de usuarios activos, distribución de usuarios por instituciones y de uso (accesos) por servicio ofrecido.

Integrar usuarios en el proceso de desarrollo: Se definirán metodologías y material para el contacto con usuarios. Además se documentarán los resultados de tales interacciones indicando la población encuestada, la distribución de respuestas y tendencias encontradas.

Diseño de adaptación de U-Cursos a móviles: Este diseño considerará los aspectos técnicos recopilados y las opiniones recogidas de los usuarios encuestados. Se generará una definición clara y justificada de las técnicas a utilizar en la implementación de la aplicación móvil junto con los requisitos funcionalidades y no-funcionales mínimos aceptables.

Especificación de API pública: Se especificarán los requisitos para la implementación de una API pública y se definirá la arquitectura para el acceso a los datos. Según el desarrollo de los objetivos anteriores se priorizarán los accesos a los datos más relevantes.

Implementación aplicación móvil: Se implementará una aplicación móvil básica que utilice las APIs definidas y cumpla con los requisitos más relevantes para los usuarios. Esto servirá para validar el cumplimiento de las metas definidas.

1.3. Metodología

Para el desarrollo de este trabajo, seguiremos los siguientes pasos

Estudio de la plataforma U-Cursos: revisar la situación actual de sus servicios y analizar los mas utilizados en la actualidad. También obtener datos de carga de sistema.

Confección encuesta usuarios: Coordinaremos una encuesta a los usuarios de U-Cursos para saber sus necesidades en una versión móvil.

Análisis resultados: Se revisarán los resultados de la encuestas y los datos obtenidos del estudio de U-Cursos actual.

Propuesta de funcionalidades: Se realizarán maquetas de la aplicación base y las funcionalidades más apropiadas para dispositivos móviles.

Diseño aplicación móvil: Se diseñará una aplicación móvil que contemple las necesidades de ADI y de sus usuarios recolectados.

Definición de APIs requeridas: Una vez diseñada la aplicación, definiremos la arquitectura requerida por parte de los servidores de U-Cursos y los servicios que se ofrecerán.

Implementación APIs: Se implementarán los accesos a datos requeridos tanto en el servidor y un cliente consumidor de ellos.

Implementación aplicación móvil: Se implementará el primer prototipo de la aplicación móvil de U-Cursos.

Capítulo 2

Estado del arte

Hemos realizado un estudio de las tecnologías que será necesario involucrar en el desarrollo de las tareas para lograr los objetivos del proyecto. Esto nos permitirá tomar una decisión objetiva y que se ajuste a las necesidades de ADI y los usuarios de U-Cursos.

2.1. U-Cursos

U-Cursos [3, 4] es un servicio desarrollado y mantenido por el Área de Desarrollo de Infotecnologías (ADI). ADI pertenece a la Facultad de Ciencias Físicas y Matemáticas (FCFM) de la Universidad de Chile. U-Cursos es una herramienta de apoyo a la docencia presencial que brinda a estudiantes y profesores de servicios que permiten una gestión ágil de los cursos utilizando tecnologías de la información. Los servicios actuales de U-Cursos se detallan a continuación:

Acta de notas Corresponde al acta final del curso, que reúne las notas del semestre y muestra la nota final del curso.

Blog Permite crear una publicación en línea donde podrá plantear ideas, reflexiones o análisis respecto del avance de su curso o de algún tema en particular.

Calendario Da la posibilidad de planear y administrar la agenda diaria, semanal y mensual de un curso.

Datos del Curso Permite ver una descripción de los datos principales del curso, tales como objetivos, evaluaciones, horario, etc.

Encuestas Permite contestar encuestas docentes.

Enlaces Da la posibilidad de publicar enlaces a sitios web que tengan relación con un curso.

Foros Cursos Da la posibilidad tanto a docentes como a alumnos de enviar mensajes en un mismo lugar para discutir temas de un curso.

Foros institucionales y de comunidades Da la posibilidad tanto a los usuarios de enviar mensajes en un mismo lugar para discutir temas de una comunidad.

Historial del Curso Muestra la actividad de un curso generada en los otros servicios.

Subir Material Da la posibilidad de subir material en el contexto de un curso, para que luego se pueda descargar por los participantes.

Notas Parciales Permite publicar notas parciales de las actividades evaluadas de un curso. Permite definir reglas complejas de cálculo de la nota final.

Novedades Da la posibilidad a los docentes de publicar noticias importantes del curso.

Tareas Permite al docente publicar enunciados de tareas. Además permite a los alumnos entregarla como un archivo que se sube al sistema. El docente puede definir parámetros de aceptación de entrega y luego bajar las entregas en masa.

Votaciones Permite realizar votaciones donde pueden participar todos los integrantes de un curso.

2.1.1. Arquitectura de la información

U-Cursos posee los siguientes elementos básicos:

Curso Representa un curso en un semestre, departamento y facultad definido.

Usuario Es un actor que puede efecutar acciones en el sistema. Un usuario puede tener distintos roles que pueden darle atribuciones adicionales como la administración de un curso o comunidad.

Institución Representa a una institución de la Universidad a la que pertenece un usuario, por ejemplo la Facultad de Ciencias Físicas y Matemáticas.

Comunidad Es un grupo que es organizado por usuarios con intereses comunes, la administración del grupo se entrega a los usuarios coordinadores de éste.

Servicio Corresponde a una funcionalidad asignada a un curso, institución o comunidad. Corresponde a los foros, material docente, acta de notas, etc.

2.1.2. Estadísticas de uso

De acuerdo a los datos de la Universidad de Chile [1], para el primer semestre de 2012 U-Cursos tenía 28.328 usuarios activos, de los cuales 5.781 son profesores y 22.547 son alumnos.

[1] http://www.uchile.cl/portal/presentacion/vicerrectoria-de-asuntos-economicos-y-gestion-institucional/convenio-de-desempeno/presentacion/88769/informe-de-gestion-u-cursos-2012-1

De acuerdo al mismo informe, la Tabla 2.1 muestra que el uso por facultad esta liderado por la Facultad de Ciencias Físicas y Matemáticas, seguida por la Facultad de Ciencias Forestales y de la Conservaciones de la Naturaleza y Facultad de Derecho.

Lugar	Unidad Académica	Puntaje
1	Facultad de Ciencias Físicas y Matemáticas	100,00
2	Facultad de Ciencias Forestales y de la Conservación de la Naturaleza	34,5
3	Facultad de Derecho	33,9
4	Facultad de Ciencias Químicas y Farmacéuticas	20,6
5	Programa Académico de Bachillerato	20,3
6	Facultad de Arquitectura y Urbanismo	18,9
7	Facultad de Medicina	13,9
8	Facultad de Ciencias Veterinarias y Pecuarias	12,4
9	Instituto de Asuntos Públicos	12,4
10	Facultad de Ciencias	11,7
11	Instituto de Nutrición y Tecnología de los Alimentos	9,6
12	Instituto de Estudios Internacionales	9,1
13	Facultad de Odontología	8,9
14	Instituto de la Comunicación e Imagen	8,1
15	Facultad de Filosofía y Humanidades	4,4
16	Facultad de Ciencias Sociales	3,3
17	Facultad de Artes	1,7
18	Facultad de Ciencias Agronómicas	0,6
19	Facultad de Economía y Negocios	0,1

Tabla 2.1: Ranking según nivel de actividad ponderado por matrícula

Del mismo reporte se aprecia que el servicio más utilizado corresponde al Foro, seguido por el servicio de publicación de material docente y general y en cuarto lugar las notas parciales (Ver Figura 2.1).

2.1.3. Accesibilidad

U-Cursos funciona como una plataforma web que se encuentra disponible en `http://www.u-cursos.cl`, donde sus usuarios ingresan con su nombre de usuario y contraseña respectivos. Dentro del portal se muestran accesos a los datos personales del usuario, sus cursos activos, las comunidades subscritas y las instituciones a las que pertenece. La Figura 2.2 muestra una vista típica desde un navegador de escritorio del portal U-Cursos con un usuario ya ingresado. La navegación es controlada por un menú vertical al lado izquierdo que permite acceso directos a los distintos canales disponibles, como cursos, comunidades e instituciones. Además, existe un menú horizontal en la parte superior del sitio que permite navegar por los distintos servicios de un curso, comunidad o institución. En la sección central de la vista se muestra el contenido del servicio al que se esta accediendo.

Respecto al acceso para dispositivo de pantalla pequeña, como teléfonos móviles, no existen

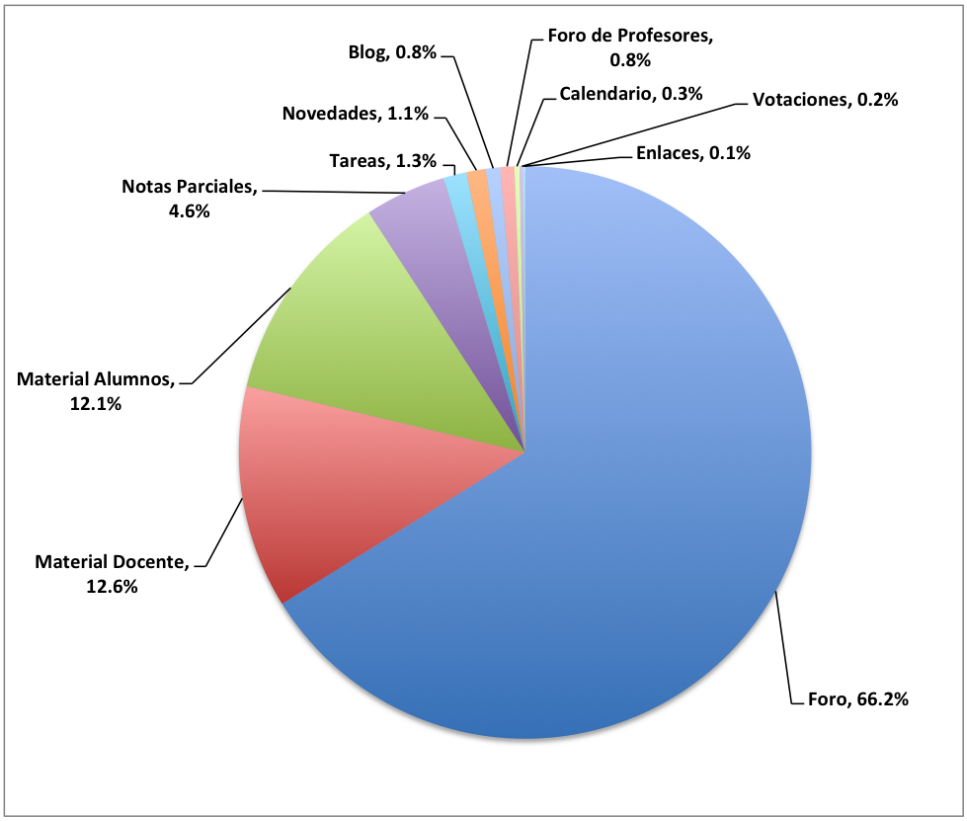

Figura 2.1: Uso de U-Cursos por servicios.

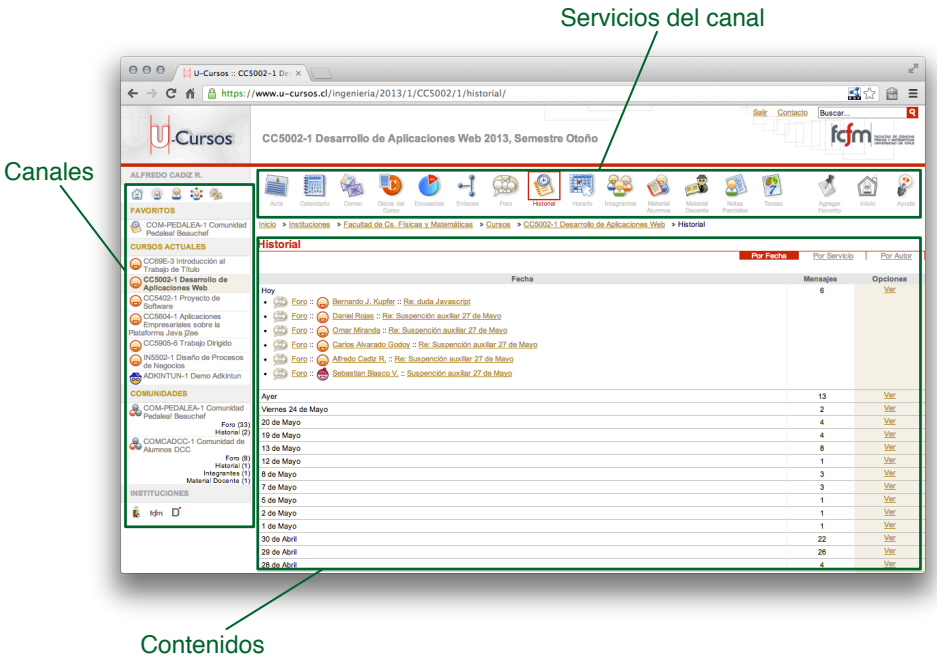

Figura 2.2: Vista de usuario ingresado a U-Cursos

vistas especializadas. Como se puede ver en la Figura 2.3, al acceder al portal desde un navegador móvil se muestra la misma vista que para computadores de escritorio o tablets. Esto provoca un exceso de información con respecto al tamaño de la pantalla, impactando en

la facilidad de navegación del sitio, ya que fuerza al usuario realizar constantes acercamientos y alejamientos a las distintas secciones de la página.

Figura 2.3: Vista de usuario ingresado a U-Cursos a través de un navegador móvil

2.1.4. Apertura de datos

Al momento de estudiar la plataforma U-Cursos se ofrecía acceso a los servicios únicamente a través del portal web `http://www.u-cursos.cl`. Esto significa que cualquier servicio derivado de los datos presentes en el sistema tiene que ser desarrollado internamente por ADI o al menos dar soporte directo a las personas interesadas en acceder a ellos. Incluso con las limitantes mencionadas, existen algunos servicios hechos por desarrolladores externos que ofrecen servicios sobre U-Cursos a los usuarios. Por ejemplo existe U-Info [2] que permite mostrar los horarios y salas actualizados para los cursos de los alumnos, además notifica ante cambios importantes que se produzcan en tiempo real. Esto muestra que existe el interés y el potencial para desarrollar complementos que mejoran la experiencia de la plataforma.

Si bien el acceso principal de los datos es a través de las vistas html, existe entre los desarrolladores de U-Cursos la voluntad de generar APIs para aplicaciones externas. Actualmente existe un punto de acceso para administrar las notificaciones push para dispositivos móviles (`https://www.u-cursos.cl/api/0/usuario/_/mis_canales/`). Este servicio permite registrar o dar de baja una llave de notificación. Además permite listar los servicios que emiten notificaciones y es posible seleccionar si se desean recibir notificaciones desde ellos o no. Los mensajes comunicados son en formato JSON, por lo que pueden ser procesados por cualquier cliente REST.

[2] http://u-info.cl/play

Lo anterior muestra que la implementación de puntos de acceso a datos de los servicios de U-Cursos es técnicamente posible. Es necesario continuar con la definición de puntos de acceso a datos para servicios de U-Cursos. Para ello debemos estudiar los que son más relevantes y definir qué datos se expondrán.

2.1.5. Autenticación de usuarios

Actualmente existe un sistema de autenticación de usuarios desarrollado por la Universidad de Chile, éste es llamado U-Pasaporte [3]. Este servicio provee una manera simple de autenticar usuarios de la Universidad de Chile para que accedan a distintos servicios en línea que ofrece la institución. De esta forma se resguarda la seguridad de los usuarios al entregar *tokens* a sitios que necesiten verificar identidad, en lugar de transmitir sus credenciales. Este servicio esta concebido para ser integrado en cualquier proyecto interesado en autenticar usuarios de la Universidad. U-Cursos hace uso intensivo de U-Pasaporte para proteger la información de los usuarios y asegurarse de presentar los servicios correspondientes a sus roles.

U-Pasaporte funciona a través de *tokens* y *superTokens* para generar requerimientos autenticados a los servicios. El *token* es un string (cadena de caracteres) que debe ser agregado como una cookie en cada requerimiento a algún servicio que soporte U-Pasaporte. Los tokens tienen un tiempo de vida limitado y deben ser actualizados frecuentemente. Para ello usuario tiene también un *superToken* que sirve para renovar los tokens que han expirado. El flujo de uso de U-Pasaporte se muestra en la Figura 2.4.

Este sistema es similar a los estándares de autenticación como OAuth [10] u OpenID [13] ya que define un protocolo de manejo de tokens para realizar requerimientos y para actualizarlos. También provee la posibilidad de invalidar una sesión en caso de problemas de seguridad.

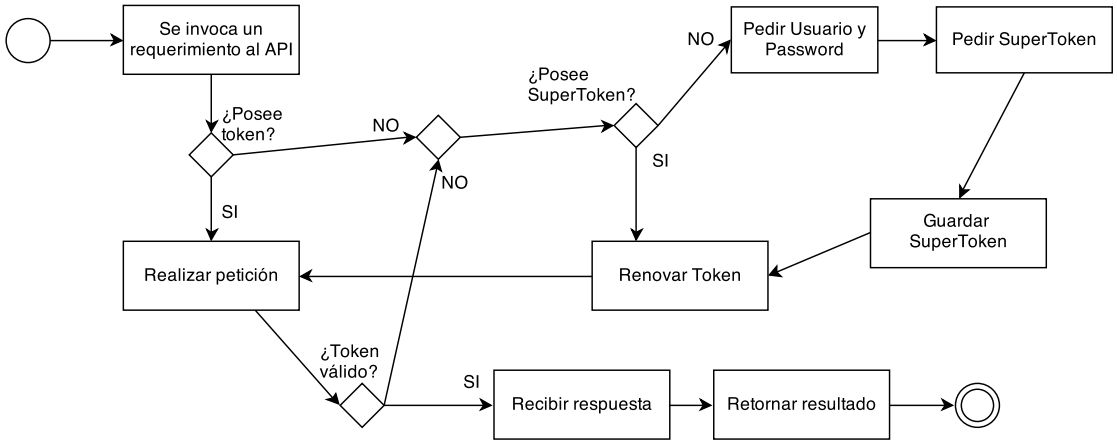

Figura 2.4: Flujo de la interacción con U-Pasaporte

[3] https://www.u-cursos.cl/upasaporte/

2.2. Implementación de APIs

La apertura de los datos de U-Cursos debe realizarse a través de APIs (Interfaces de programación de aplicaciones) que permitan ofrecer a sus usuarios un acceso homogéneo y transparente. Esto permite una mejor adopción y una integración más directa con aplicaciones utilicen dichos datos. Para ello existen distintos modelos que permiten ofrecer APIs web, los más importantes actualmente son SOAP y REST. Ambos modelos son bien conocidos y poseen herramientas que permiten integrar rápidamente las aplicaciones con ellos. A continuación presentamos ambas tecnologías observando sus ventajas y desventajas.

2.2.1. Simple Object Access Protocol

Simple Object Access Protocol (SOAP) [1] es un protocolo de intercambio de información de manera descentralizada y distribuida. Inicialmente fue propuesta por Microsoft, actualmente el estándar es mantenido por el XML Protocol Working Group de la World Wide Web Consortium (W3C).

Tiene como objetivo ser un protocolo simple y extensible. Además es independiente del medio de transporte por lo que puede ser implementado sobre HTTP o cualquier otro protocolo de transferencia. Esto permite la interoperabilidad con sistemas legados que utilicen protocolos distintos al HTTP.

La Figura 2.5 muestra la estructura de un mensaje SOAP. Esta compuesto por un encabezado (header) y un cuerpo (body), ellos están contenidos dentro de un sobre (envelope). Estos tres componentes permiten definir el tipo de mensaje a enviar, las reglas de codificación para el mensaje y las consultas/respuestas en un lenguaje común.

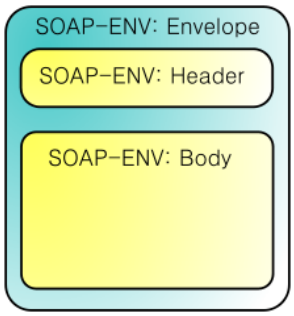

Figura 2.5: Estructura de un mensaje SOAP

SOAP utiliza XML como lenguaje para describir todos sus componentes. Esto le permite ser extensible y tecnológicamente neutral. El código en el Listing 2.1 muestra un mensaje básico en SOAP.

Listing 2.1: Mensaje básico SOAP

```
1 POST /InStock HTTP/1.1
2 Host: www.example.org
3 Content-Type: application/soap+xml; charset=utf-8
```

```
4  Content-Length: 299
5  SOAPAction: "http://www.w3.org/2003/05/soap-envelope"
6
7  <?xml version="1.0"?>
8  <soap:Envelope
9          xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
10         <soap:Header>
11         </soap:Header>
12         <soap:Body>
13                 <m:GetStockPrice
14                         xmlns:m="http://www.example.org/stock">
15                         <m:StockName>IBM</m:StockName>
16                 </m:GetStockPrice>
17         </soap:Body>
18 </soap:Envelope>
```

SOAP es criticado por ser aparentemente complejo en comparación con REST. Además sus mensajes en XML pueden ser mucho mas grandes que los de otros protocolos similares, como CORBA. Además, su alto poder de expresividad puede ser mal utilizado y generar mensajes que rompan el protocolo o violen la neutralidad tecnológica de los mensajes [12].

2.2.2. REST

REpresentational State Transfer (REST) [8] es un estilo de arquitectura de software para construir sistemas distribuidos escalables. Fue desarrollado por el Technical Architecture Group (TAG) de la W3C en paralelo con HTTP/1.1.

REST esta basado en 4 principios fundamentales:

Identificación de recursos con URIs: Los recursos se representan por URIs (Identificadores de Recurso Universales) que son los puntos donde los clientes interactúan con el sistema.

Interfaz uniforme: Los recursos son manipulados utilizando cuatro operaciones o verbos: GET, DELETE, PUT, POST. Cada una de las operaciones tiene un modo de funcionar bien definido.

Mensajes autodescriptivos: Cada mensaje incluye suficiente información para describir como se debe procesar.

Interacciones con estado a través de hipervínculos: Las interacciones no tienen estado. Para crear interacciones con estados, estos deben transmitirse explícitamente.

Al ser un estilo de arquitectura y no un protocolo, no existe un estándar oficial para REST. Existen actualmente 2 recomendaciones: Hi-REST y Lo-REST.

Hi-REST: Recomienda utilizar todos los verbos: GET, POST, PUT y DELETE. Además las URIs deben ser uniformes y con un formato adecuado. Se recomienda el uso de XML para los mensajes, aunque el uso de otros formatos como JSON o YAML también se

esta considerando.

Lo-REST: Minimizar el uso de verbos a sólo GET y POST. Donde GET es para peticiones seguras (sin cambios en el servidor) y POST para todo lo demás.

REST es considerado simple ya que utiliza estándares de la W3C que son bien conocidos, como HTTP, XML, URI, MIMEs y la arquitectura requerida se encuentra disponible casi universalmente. Por otro lado, al no ser un estándar, no hay reglas claras de cómo se debe implementar un servicio RESTful, sino que recomendaciones. Además, en algunos casos, la regla de GET para peticiones seguras no se puede cumplir dado que tales peticiones tienen un largo máximo y hay que utilizar otros verbos para ese tipo de consultas [12].

2.3. Implementación de aplicaciones móviles

Los teléfonos inteligentes actuales poseen una gran poder de procesamiento y almacenamiento en comparación a generaciones anteriores. Esto les ha permitido mejorar el desempeño de tareas como la navegación en internet, permitiendo el despliegue de páginas ricas en contenido e interacciones. Además tales dispositivos aparte de alojar sus aplicaciones esenciales, puedan instalar programas nativos desarrollados por entidades externas al fabricante del móvil o el sistema operativo. Esto les ha brindado una gran flexibilidad en la forma que pueden desplegar aplicaciones, pero también le agrega un nuevo punto de decisión al desarrollador. En esta sección analizaremos las actuales alternativas para desarrollar aplicaciones para móviles.

2.3.1. Aplicación web

Inicialmente los navegadores en dispositivos móviles eran capaces de desplegar contenido HTML 4.01 muy simple y con pocas funcionalidades. Esto indujo a que el desarrollo para dispositivos móviles se limitase a ofrecer vistas reducidas y simples en comparación a las versiones para web de escritorio eran ricos en contenidos e interacciones.

Esto cambió con la nueva generación de teléfonos inteligentes, los cuales gracias a su mayor capacidad de procesamiento y almacenamiento pueden incorporar navegadores funcionando con el mismo motor de renderizado que sus versiones de escritorio, dando soporte a tecnologías actuales del desarrollo web como HTML5, CSS3 y JavaScript. Esto ha permitido que existan servicios basados en la web que pueden ser desplegados en dispositivos móviles, conservando las funcionalidades de su versión para escritorio [15]. Los ejemplos más notables son actualmente los sitios móviles de Facebook[4] y Twitter[5].

A pesar de los avances en el despliegue de sitios web móviles y sus capacidades actuales, esta tecnología posee algunas restricciones. Primero, al no ser código nativo, sino que interpretado en tiempo de ejecución, el despliegue de gran cantidad de información o de gráficos

[4] `https://m.facebook.com`
[5] `https://mobile.twitter.com`

complejos tiene un impacto importante en la rapidez de la aplicación que puede afectar la calidad de la experiencia de uso. Además, el acceso a las características internas del dispositivo (cámara, GPS, NFC, etc.) móvil son limitadas y depende del fabricante si son ofrecidas o no. Adicionalmente, al ser servicios alojados en servidores externos, es obligatorio tener acceso a internet para poder utilizarlos. Finalmente, el diseño desplegado no será acorde al *look&feel* nativo del dispositivo sino que será homogéneo sin importar desde donde se acceda. Aunque esto último no es un defecto, es importante a considerar al momento de diseñar las interfaces de la aplicación.

2.3.2. Aplicación nativa

El desarrollo de aplicaciones nativas para teléfonos inteligentes se ha incrementado drásticamente en los últimos años gracias a iOS de Apple y Android de Google. Sus entornos de desarrollo robustos y bien documentados han permitido a los desarrolladores implementar rápidamente aplicaciones que aprovechan las capacidades de los dispositivos donde se alojan [9].

Desarrollar una aplicación nativa tiene ventajas en el desempeño de ésta: al ser código compilado la ejecución está optimizada y puede ejecutar procesos pesados aprovechando al máximo el hardware. Además las aplicaciones nativas tienen acceso completo a las APIs del dispositivo, permitiendo explotar todas las funcionalidades que ofrece, como sensores, acceso a archivos y coordinación con otras aplicaciones. Adicionalmente las aplicaciones nativas pueden funcionar sin estar necesariamente conectadas a internet. Por parte de la interfaz de usuario, una aplicación nativa se beneficia de los elementos gráficos provistos por el sistema operativo móvil, permitiendo que la experiencia sea uniforme con respecto a otras aplicaciones.

Por otro lado, desarrollar para cada dispositivo disponible puede ser muy costoso. Esto es porque cada fabricante posee su propio entorno de desarrollo y en muchos casos es necesario crear la aplicación desde cero para cada uno, con el alto costo que tiene mantener cada proyecto por separado. Además, en algunos casos el fabricante puede imponer obstáculos que podrían considerarse arbitrarios para la publicación de aplicaciones en sus tiendas.

2.3.3. Aplicación nativa multiplataforma

Como hemos visto, tanto el desarrollo web móvil como el nativo poseen beneficios y desventajas. Aunque HTML5 ha avanzado en el acceso a ciertos componentes nativos (por ejemplo, GPS) y al ofrecer interfaces más atractivas, aún es necesaria una mayor madurez de la tecnología para que pueda reemplazar a una aplicación nativa [5, 15].

El desarrollo multiplataforma para dispositivos móviles busca facilitar el desarrollo de aplicaciones nativas en móviles. Esto se logra abstrayendo al desarrollador de los entornos de desarrollo de cada dispositivo y su lugar ofrece implementar la lógica y vistas de la aplicación en un lenguaje común, normalmente HTML5 + CSS3 + JavaScript. Esto es implementado a

través la generación de aplicaciones móviles que despliegan los contenidos en una vista web incrustada desplegando html local. Adicionalmente se ofrecen formas de acceder a las APIs del dispositivo como lo haría una aplicación 100 % nativa.

Los existen variados frameworks de desarrollo multiplataforma como PhoneGap[6], Titanium[7], Xamarin[8] y Oracle ADF Mobile[9], entre otros. Cada uno ofrece distintas funcionalidades y distintos términos de distribución, por lo que los desarrolladores interesados pueden usar el que mejor les acomode.

El desarrollo en frameworks multiplataforma tienen la ventaja de proveer una base común de código fuente que luego se traduce en proyectos nativos para los distintos sistemas operativos móviles. Además los costos de desarrollo son menores ya que las tecnologías web son mucho más populares que las de los entornos nativos de cada dispositivo y es más fácil conseguir programadores hábiles. Además los costos de mantención de las aplicaciones bajan ya que los cambios se propagan automáticamente a todos los dispositivos al re-compilar los proyectos.

Hay que considerar que este tipo de aplicaciones tienen algunas desventajas del desarrollo web: la lógica al ser ejecutada localmente a través de JavaScript tiene un desempeño menor que el código nativo, normalmente no es un problema si son aplicaciones pequeñas, pero si ejecutan algoritmos pesados o hacen uso intensivo de gráficos, como en el caso de los juegos, pueden presentar problemas. Adicionalmente, cuando existe comunicación con APIs nativas a veces se requiere de implementaciones ad-hoc en el lenguaje del dispositivo que rompería el esquema de una base de código común. Finalmente las interfaces gráficas e interacciones son comunes a todos los dispositivos por lo que se puede perder la sensación de estar en una aplicación nativa si no se toman buenas decisiones al momento de diseñar las vistas.

2.3.4. ¿Qué tecnología usar?

Todas las alternativas mencionadas anteriormente son igualmente válidas. Entonces, la selección de tecnología a usar para el desarrollo de una aplicación depende fuertemente del interés del equipo desarrollador. Algunos aspectos que deben revisarse al evaluar el tipo de implementación son:

Uso de APIs nativas: Es necesario evaluar si se requiere utilizar características exclusivas de los dispositivos que servirán de clientes. En el caso de necesitar acceso componentes especiales será necesario realizar implementaciones ad-hoc para el dispositivo.

Dispositivos objetivo: Se implementará sólo en un sistema operativo en particular o se desea abarcar la mayor cantidad posible.

Desempeño esperado: La aplicación despliega información simple en listas o necesitará

[6] `http://phonegap.com/`
[7] `http://www.appcelerator.com/titanium/`
[8] `http://xamarin.com/`
[9] `http://www.oracle.com/technetwork/developer-tools/adf-mobile/overview/index.html`

gran poder de procesamiento con algoritmos pesados.

Diseño y experiencia de usuario: Se desea conservar el *look&feel* propio del sistema que desplegará o si se prefiere una interfaz uniforme para todas las instancias de la aplicación.

Capítulo 3

Análisis

En este capítulo detallamos el trabajo previo a la implementación. Este trabajo consistió en la ejecución y revisión de una encuesta a los usuarios de U-Cursos. La encuesta tiene como objetivo recopilar la opinión de los usuarios del sistema sobre las expectativas de una versión móvil de U-Cursos. Junto con lo anterior, hemos analizado el tráfico del servidor web la plataforma para conocer la actividad de los accesos desde móviles. Esto permitirá guiar el diseño y la implementación de las APIs y la aplicación móvil hacia características de software importantes para los usuarios.

3.1. Encuesta a usuarios

Durante el semestre de Otoño 2013, hemos diseñado una encuesta a los usuarios de U-Cursos. El motivo de esta encuesta es estudiar el uso de teléfonos inteligentes entre sus usuarios. Además nos servirá para analizar el interés de los usuarios por un cliente móvil de la plataforma. La encuesta se dividió en 3 secciones: perfilamiento, servicios actuales y servicios nuevos.

En primer lugar, se realizaron preguntas para perfilar a los usuarios (Figura 3.1). Se les consultó el año de la carrera que cursan. Además se pidió el tipo de teléfono que poseen. Para la lista de opciones posibles se prefirió destacar a los teléfonos con sistemas operativos móviles más actuales, porque a ellos apuntará el desarrollo de la aplicación móvil. Por lo tanto, las opciones presentes fueron: *Android*, *Blackberry*, *iPhone*, *Windows Phone*, *Otro tipo* o *No tiene*. Adicionalmente se le ofreció al usuario la opción de ser contactado para una posible marcha blanca de la aplicación. En ese caso tiene la opción de dar su correo electrónico para ser contactado.

Además, algunos datos de perfilamiento fueron recolectados automáticamente: *User-agent* del cliente HTTP con el que se contestó la encuesta y las *facultades* a la que pertenece el usuario. Esto gracias a que los usuarios debían contestar la encuesta en durante una sesión activa de la plataforma.

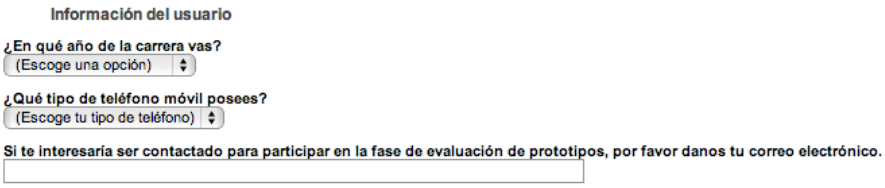

Figura 3.1: Encuesta U-Cursos móvil: perfilamiento

En la segunda parte de la encuesta se consultó sobre los servicios actuales que los usuarios necesitan en una versión móvil. Para ello se le presentó al usuario una matriz donde debe dar relevancia a los servicios actuales de U-Cursos en una posible versión móvil (Figura 3.2). Se presentaron los servicios listados en la primera columna de la matriz y luego 5 opciones desde *Nada importante* hasta *Muy importante* de las cuales sólo podía elegir una.

Servicios Actuales

Actualmente ¿Qué tan importantes son los siguientes servicios al momento de utilizar U-Cursos en tu teléfono móvil?
Si el servicio no se ofrece en tu facultad, escoge "Sin opinión".

	Nada importante	Poco importante	Sin opinión	Algo importante	Muy importante
Acta de notas	○	○	○	○	○
Blog	○	○	○	○	○
Calendario	○	○	○	○	○
Datos del Curso	○	○	○	○	○
Encuestas	○	○	○	○	○
Enlaces	○	○	○	○	○
Foros Cursos	○	○	○	○	○
Foros institucionales y de comunidades	○	○	○	○	○
Historial del Curso	○	○	○	○	○
Subir Material	○	○	○	○	○
Notas Parciales	○	○	○	○	○
Novedades	○	○	○	○	○
Tareas	○	○	○	○	○
Votaciones	○	○	○	○	○

Figura 3.2: Encuesta U-Cursos móvil: servicios actuales

Finalmente se le plantearon al encuestado 3 nuevos servicios que son factibles de implementar en un dispositivo móvil (Figura 3.3): *Encontrar la sala* con GPS, *Notificaciones Push* y *subir material* desde el móvil. Además se presentó una caja de texto libre para que puedan comentar sobre servicios que no han mencionado en la encuesta.

3.1.1. Resultados

La encuesta fue ejecutada durante la primera semana de Septiembre de 2013 a través de un mensaje que se desplegó a los usuarios de U-Cursos. El mensaje permitía contestar o ignorar la encuesta según el interés del usuario (Ver Figura 3.4). La encuesta estuvo disponible por una semana para todos los usuarios de la plataforma, en este lapso se recibieron 4806 respuestas de las cuales la mayoría provienen desde la Facultad de Ciencias Físicas y Matemáticas con un 47 % de los registros, seguido por la Facultad de Medicina (14 %) y la Facultad de Derecho (13 %) (Figura 3.5). Respecto a la distribución por año de la carrera (Figura 3.6), ésta nos muestra que la mayoría respuestas se concentran en los primeros 4 años de carrera para luego disminuir hacia años avanzados y personas ya tituladas. También hemos analizado la

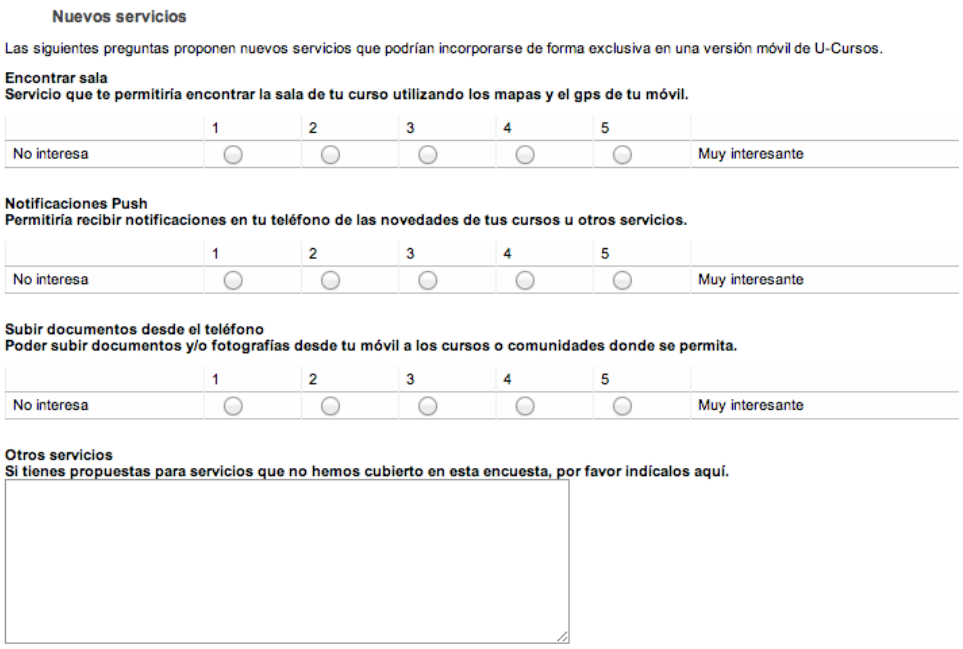

Figura 3.3: Encuesta U-Cursos móvil: nuevos servicios

distribución de los tipos de dispositivo móvil declarado por los encuestados. La Figura 3.7 nos muestra que la mayoría de los usuarios poseen teléfonos Android, seguidos por los usuarios de dispositivos iOS. Para teléfonos inteligentes con otros sistemas operativos móviles, la cantidad de respuestas es mucho menor.

Figura 3.4: Propuesta para contestar la encuesta en U-Cursos

Con respecto a las respuestas recibidas, primero nos enfocamos en conocer los servicios actuales más requeridos por los usuarios. El gráfico en la Figura 3.8 muestra que los servicios de *Notas Parciales*, *Novedades*, *Subir Material*, *Tareas*, *Foros de Cursos* y *Votaciones* son los que mayor interés generan. Luego hay otro grupo donde las preferencias están balanceadas: *Calendario*, *Datos de Curso*, *Encuestas*, *Enlaces*, *Foro de instituciones* y el *Historial*. Estos datos nos permitirán priorizar y ordenar el desarrollo de los módulos que se adapten a móviles.

Con respecto a las nuevas funcionalidades para dispositivos móviles, observamos que hay una buena recepción para todos los servicios propuestos. La Figura 3.9 muestra que la opción *muy interesante* es claramente la más escogida en todos los servicios propuestos. De estos servicios se ve una mayor preferencia por la geolocalización de las salas de clases, seguido de

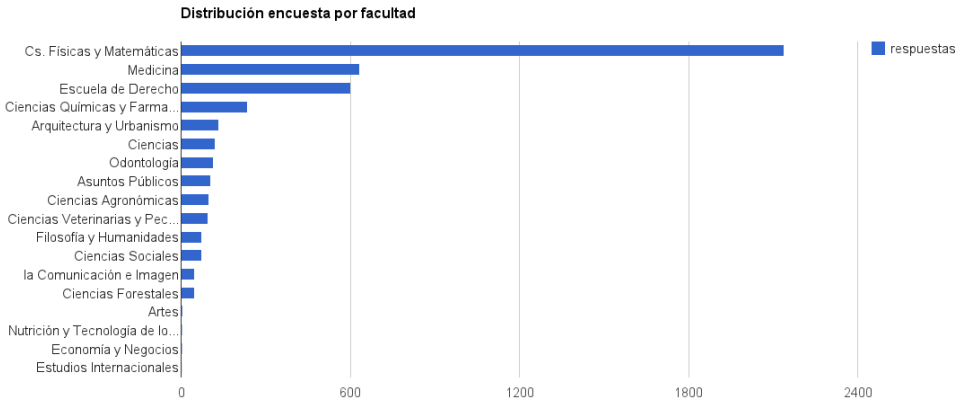

Figura 3.5: Distribución encuesta entre las facultades

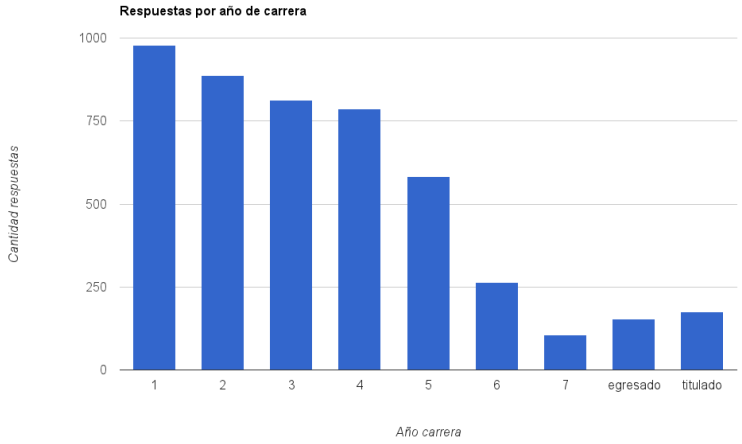

Figura 3.6: Distribución encuesta por año de la carrera

subida de archivos desde los móviles y notificaciones push.

Finalmente, hemos analizado las respuestas entregadas por los usuarios en la caja de texto libre. A pasar que se ofreció para comentar sobre funcionalidades que no se consideraron en la encuesta, los usuarios lo utilizaron para fines varios, como felicitaciones o recalcar su interés en características que les son muy relevantes aunque se salgan del ámbito móvil. De un total de 4806 encuestas contestadas, 580 incluían información en la caja de texto.

Hemos clasificado en 3 categorías: *Servicios actuales*, *Servicios nuevos móvil* y *Servicios nuevos general*, además hemos ordenado los comentarios según su frecuencia.

La Figura 3.10 muestra cuales son los conceptos más relevantes respecto a los servicios actuales de U-Cursos. Se puede apreciar los más mencionados son el *Horario personalizado*, acceso a los *correos* del cuerpo docente, las *notificaciones push*, acceso al *material docente*, integración de los eventos en *calendario* del móvil y las *notas parciales*.

Por otro lado, la Figura 3.11 muestra los conceptos que han aparecido en la encuesta

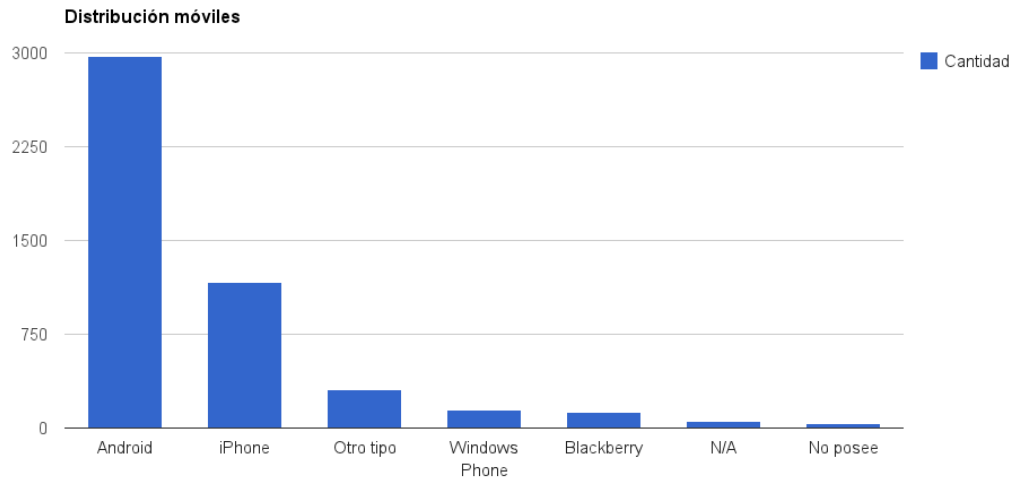

Figura 3.7: Uso de U-Cursos por sistema operativo.

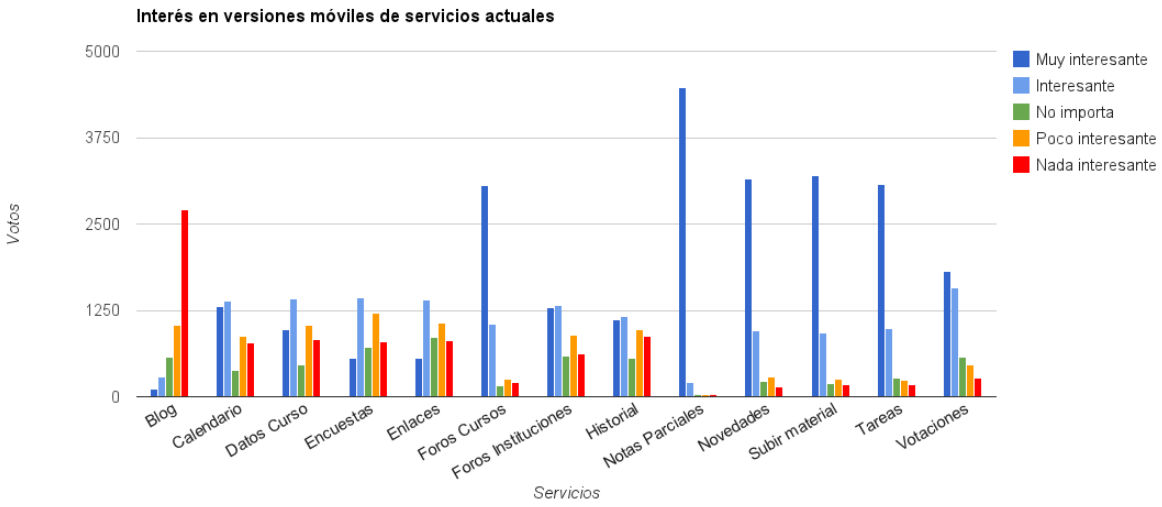

Figura 3.8: Interés por servicio actual

relacionados a servicios móviles. Se destaca la necesidad de seleccionar los canales para notificaciones push *(push granular)*, además existe interés por al menos *previsualizar material* docente presente en un curso. Se destacó además la posibilidad de *encontrar salas* a través de la funcionalidad de GPS del dispositivo móvil. Adicionalmente se pidieron aspectos de calidad como una interfaz *simple* y amigable.

Como mencionamos anteriormente, el público aprovechó la encuesta para solicitar funciones más allá del alcance de un cliente móvil, sino que conciernen a toda la plataforma. Como se puede apreciar en la Figura 3.12, la solicitud con mayor interés es la de ofrecer *salas de chat* para el uso en cursos. Además integración con *bibliotecas*, que el servicio sea accesible desde más dispositivos, integración con redes sociales y planos de ubicación dentro de facultades.

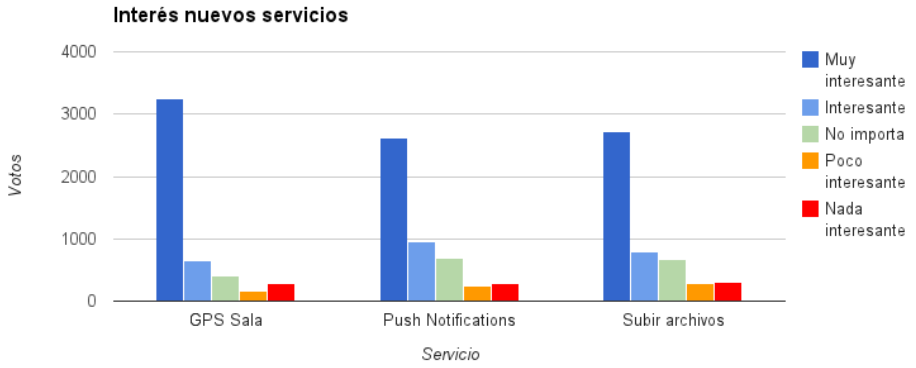

Figura 3.9: Interés por servicio actual

Figura 3.10: Conceptos más relevantes respecto a servicios actuales

Figura 3.11: Conceptos más relevantes respecto a servicios para móviles

Figura 3.12: Conceptos más relevantes respecto a nuevos servicios generales

En el Apéndice A, Tabla A.1 se muestran en detalle los conceptos y las frecuencias de éstos en las tres categorías mencionadas.

3.2. Análisis de accesos

Como complemento a la encuesta a usuarios, hemos hecho un análisis de los accesos al servidor web de U-Cursos. Esto tiene como objetivo conocer los servicios más populares y analizar el comportamiento de los usuarios que actualmente acceden desde móviles. Para ello hemos analizado la información correspondiente a Abril de 2013 de los registros de acceso (logs) del servidor Apache de la plataforma. Hemos escogido Abril como referencia ya que durante ese periodo no hay hitos académicos importantes que puedan distorsionar los registros, como por ejemplo los fines de semestre donde existe un alza importante en el acceso a las actas de notas, que no ocurre en otros momentos.

Los registros de acceso del servidor web contienen información sobre cada requerimiento que se ejecuta, incluyendo datos sobre el servicio que se accedido e información sobre el cliente. Dado esto y la gran cantidad de usuarios activos en el sistema, hemos obtenido 100Gb en archivos de registros para el periodo estudiado.

Un registro es representado como una línea de texto en un archivo de registros. Un registro descompuesto según sus componentes por saltos de línea se puede observar en el código 3.1.

Listing 3.1: Desglose de un registro de acceso a U-Cursos

```
1  [01/May/2013:10:50:01 -0400]
2  200
3  GET
4  /medicina/2013/1/MMEDLEG5/1/foro/?offset=0\&id_mensaje=3490865
5  HTTP/1.1
6  5277
7  https://www.u-cursos.cl/medicina/2013/1/MMEDLEG5/1/historial/
8  Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.31 (KHTML, like Gecko)
9  Chrome/26.0.1410.64 Safari/537.31
10 186.105.242.235
11 1
```

Los componentes son los siguientes:

1 Fecha del requerimiento.

2 Código HTTP resultante.

3 Método HTTP.

4 Ruta del requerimiento.

5 Protocolo HTTP utilizado

6 Tamaño de la respuesta en bytes, excluyendo encabezados.

7 Campo X-Forwarded-For del enbacezado HTTP.

8-9 User-agent del cliente que hizo la petición.

10 Dirección IP que originó el requerimiento.

11 Tiempo en segundos ocupado en el requerimiento.

De tales registros hemos analizado el campo *user-agent* y la *ruta del requerimiento*. El user-agent contiene información entregada por el cliente http al momento de acceder al sistema. De este modo hemos podido extraer y contabilizar información sobre el dispositivo, sistema operativo y software que se utilizó para acceder a la plataforma. La *ruta del requerimiento* indica qué recurso se accedió en el requerimiento y desde ella se puede determinar fácilmente qué servicio se esta utilizando. La Figura 3.13 muestra que el sistema operativo Windows de Microsoft es la plataforma más utilizada en sus versiones de escritorio, mientras que Linux y Mac Os X de Apple representan un acceso minoritario, pero acorde a la tendencia mundial en el uso de tales sistemas [1].

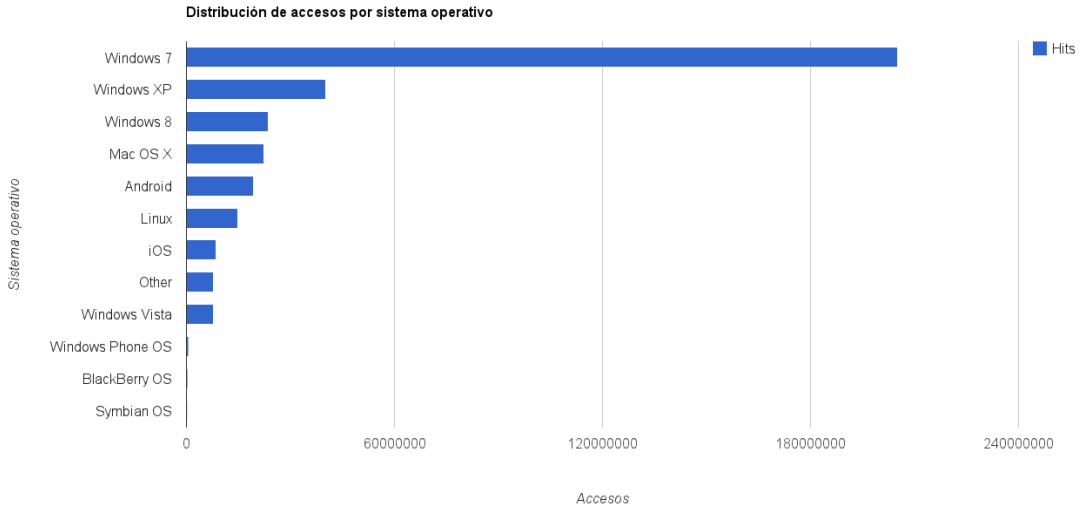

Figura 3.13: Uso de U-Cursos por sistema operativo.

Respecto a la representación de los móviles, se puede ver que el acceso desde tales dispositivos es minoritario comparado con los sistemas de escritorio, donde los dispositivos que declararon ser iOS y Android (incluyendo tablets que los utilizan) no superan el 8 % de uso. Si contrastamos estos datos con los estudios de la Subsecretaría de Telecomunicaciones de Chile, encontramos discrepancias. Esto es porque en 2012 se alcanzó una penetración de las conexiones 3G móviles de 22,11 conexiones por cada 100 habitantes, que es el doble de la internet fija con 12,18 conexiones por cada 100 habitantes. Además se indica que Chile en el segundo país con mayor crecimiento en la adquisición de dispositivos iOS y Android [7]. Estos hechos nos muestran que a pesar de la gran penetración de las conexiones móviles y adquisición de teléfonos inteligentes, los accesos a U-Cursos desde dispositivos móviles son mucho menores a los que potencialmente puede recibir.

Además, hemos analizado como se distribuyen los accesos a los servicios de U-Cursos para

[1] http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0

los dos dispositivos móviles con mayor relevancia. Esto nos permitirá complementar la información que ya tenemos de las encuestas a usuarios utilizando información objetiva desde los registros de acceso. La Figura 3.14 nos muestra que la mayor cantidad de accesos corresponden a los *foros*, seguido del *material docente* y el *historial*. También las *notas parciales*, acceso al *material alumnos* y las *novedades* se muestran con un número de accesos considerables. Es interesante notar que adicionalmente los servicios de *integrantes* de curso, *horario* y *datos curso* tienen también un número de accesos que no son despreciables, pero solamente en los usuarios de iOS, mientras que en Android la cantidad de accesos es mucho menor.

Es necesario recordar que los registros de accesos fueron analizados para el mes de Abril para evitar peaks por periodos críticos. Por ello algunos servicios como el *Acta* no tienen un impacto fuerte, lo cual para nuestro análisis es apropiado para comprender los servicios que son siempre necesarios. También cabe mencionar que el servicio *historial* es normalmente la página de inicio de los usuarios luego de ingresar correctamente sus credenciales. A causa de lo anterior, este servicio en particular tiene más accesos que los que realmente el usuario buscó realizar en su sesión desde un dispositivo móvil.

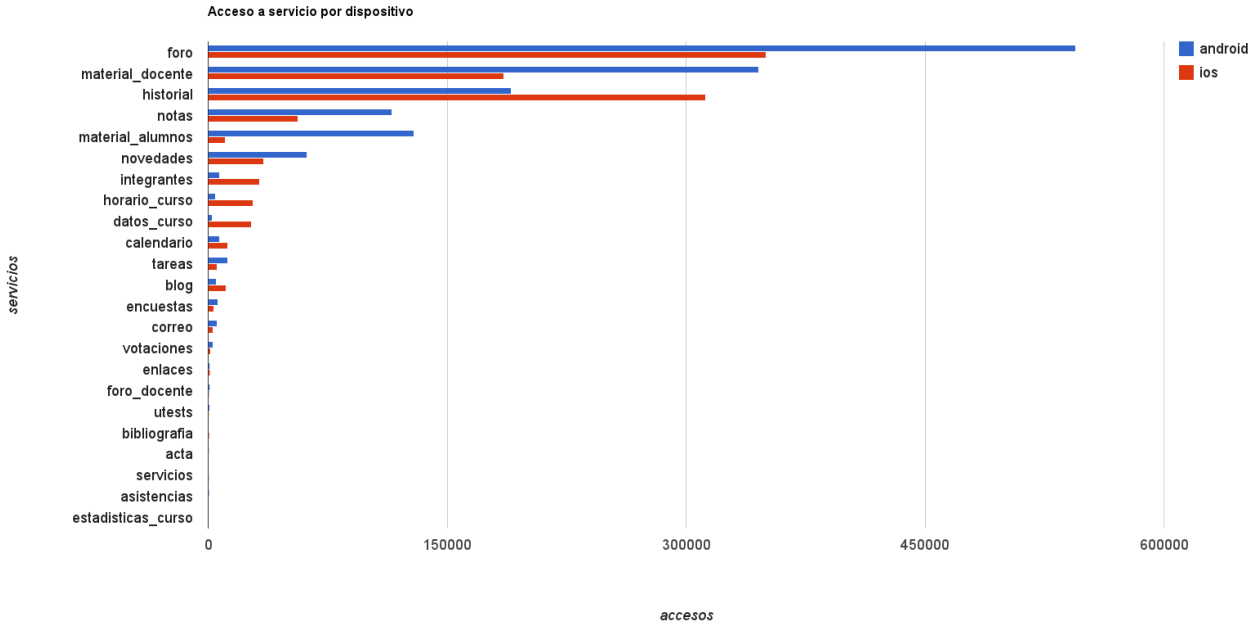

Figura 3.14: Accesos a servicios de U-Cursos por móvil

3.3. Conclusiones

En este capítulo hemos propuesto y ejecutado dos análisis a los servicios de U-Cursos. Primero confeccionamos una encuesta para evaluar las preferencias de los usuarios para las funcionalidades de una versión para teléfonos móviles. La encuesta se le presentó a los usuarios durante una semana y recibimos sobre 4000 respuestas. También realizamos un análisis de los registros de accesos del servidor web de U-Cursos. Este análisis tiene como fin determinar la proporción de accesos efectivos realizados con dispositivos móviles en la actual versión de U-Cursos y cuales son los servicios más utilizados desde ellos.

De las encuestas a usuarios hemos aprendido que hay un gran interés en mejorar el acceso móvil a la plataforma. Eso tanto en las buena recepción a las propuestas de servicios nuevos como con los comentarios en la caja de texto libre. Del análisis de los registros de U-Cursos hemos determinado que el acceso desde móviles corresponde solo al 8 % del total, lo que no coincide con los altos índices de conectividad móvil del país. Todo lo anterior nos muestra que existe la necesidad de los usuarios por acceder a sus servicios desde sus móviles, pero no se refleja en los accesos efectivos por la falta de soporte para los dispositivos.

De los estudios a los usuarios y a los registros de U-Cursos hemos confeccionado la Tabla 3.1 que ordena los servicios según cada categoría, ordenados de mayor a menor según importancia. De esta manera es fácil determinar cuáles son los servicios a los que se les debe dar mayor prioridad al momento de darle soporte para móviles.

Encuesta opciones		Registros servidor web	Encuesta texto libre	
Servicio	Nuevos	Servicios actuales	Actuales	Nuevos móvil
Notas Parciales	GPS Sala	foro	Mi Horario	Chat
Subir material	Subir archivos	material_docente	Correos	Push granular
Novedades	Push Notifications	historial	Notificaciones	Previsualizar Material
Tareas		notas	Material Docente	Buscar Sala
Foros Cursos		material_alumnos	Calendario	Simplicidad
Votaciones		novedades	Notas Parciales	Multiplataforma
Foros Instituciones		integrantes	Foro	Marcar Asistencia
Calendario		horario_curso	Dropbox	Buscar Personas
Datos Curso		datos_curso	Notas Guía	Offline
Historial		calendario	Tareas	Reservar Laboratorio
Encuestas		tareas	Acceso a cursos pasados	Línea de tiempo
Enlaces		blog	Novedades	Solicitud Certificados
Blog		encuestas	Favoritos	Widget Android
		correo	Actas	Calculadora Notas
		votaciones	BIA	Coordinador
		enlaces	Encuestas	Google Drive
		foro_docente	Enlaces	Colas de impresión
		utests	Histórico de notas	Correo Institucional
		bibliografia	Sesión persistente	Accesos rápidos
		acta	Administración Cursos	Botón +1 y -1
		servicios	Afiches	Catálogo de cursos

Tabla 3.1: Resumen análisis de los servicios de U-Cursos

Capítulo 4

Diseño

En este capítulo describiremos el diseño de los sistemas que permitirán implementar un cliente móvil de U-Cursos. Primero describiremos los requisitos para la API de acceso y aplicación móvil, luego definiremos la arquitectura del software a utilizar. Luego explicaremos el funcionamiento de las APIs de acceso a los datos de U-Cursos, finalmente detallaremos el diseño de la aplicación móvil.

4.1. Requisitos del software

Basados en el análisis en el capítulo 3 y en las necesidades de ADI hemos consolidado requerimientos para el desarrollo del cliente móvil.

Primero, es necesario considerar que U-Cursos cuenta ya con una gran cantidad de servicios a través de su servidor web. Si bien algunos servicios no tienen sentido en dispositivos móviles, la mayoría son posibles de implementar, y es deseable por ADI que la aplicación móvil sea lo más completa posible. Además, por parte de ADI y de los usuarios se ha destacado el interés por explotar las capacidades que ofrecen los dispositivos móviles, como acceso a cámara, GPS, o notificaciones push.

4.1.1. Requisitos aplicación móvil

A partir de las necesidades de ADI y los usuarios de U-Cursos hemos definido una lista de requerimientos para la aplicación móvil.

Notificaciones Push: Se requiere que la aplicación pueda recibir mensajes desde el servidor de U-Cursos para notificar en tiempo real a sus usuarios en cambios en sus canales actuales. Es necesario que el usuario pueda escoger los canales de los que desea recibir información.

Acceso todos los servicios: Es relevante determinar cómo ofrecer la mayor cantidad de los servicios actuales en el cliente móvil.

Optimizar uso del móvil: Dentro de lo posible implementar servicios de manera nativa.

4.1.2. Requisitos API

Para que una aplicación móvil pueda recibir contenidos desde los servidores de U-Cursos es necesario definir un acceso por medio de APIs. Con respecto a estas APIs se han definido los siguientes requerimientos:

Integración con U-Pasaporte: Para la autenticación de los usuarios, es necesario utilizar el sistema U-Pasaporte utilizado por la Universidad de Chile para centralizar la identificación de sus usuarios.

API extensible: Es importante considerar que los servicios pueden incrementarse.

Comunicación transparente: Utilizar formatos de datos bien documentados y utilizar nombres de recursos fáciles de comprender por los desarrolladores.

4.2. Arquitectura del software

Dados los requerimientos mencionados, hemos definido una arquitectura para integrar APIs al actual servicio de U-Cursos y para implementar las notificaciones push.

En la Figura 4.1 se muestra la arquitectura propuesta. Actualmente el servicio principal de U-Cursos se aloja en servidores web Apache (1), los datos del sistema son persistidos en bases de datos MySQL (2). Cuando el servidor web recibe requerimientos, actualmente envía contenido en formato HTML (3). Se propone que adicionalmente el mismo servidor reciba peticiones de API y responda con contenido en formato JSON[1] (4).

Para las notificaciones push se utilizará la base de datos del tipo *key/value* Redis[2] (5). Luego, una instancia de Node.js[3] (6) procesa las notificaciones pendientes y las despacha a los servicios de mensajería de los proveedores de notificaciones móviles (7).

4.3. API de U-Cursos

Como se mencionó en la sección 4.1.2, necesitamos crear una API fácilmente extensible y transparente para los desarrolladores que la utilicen. Dado que actualmente U-Cursos funcio-

[1] JavaScript Object Notation: http://www.json.org/
[2] http://redis.io/
[3] http://nodejs.org/

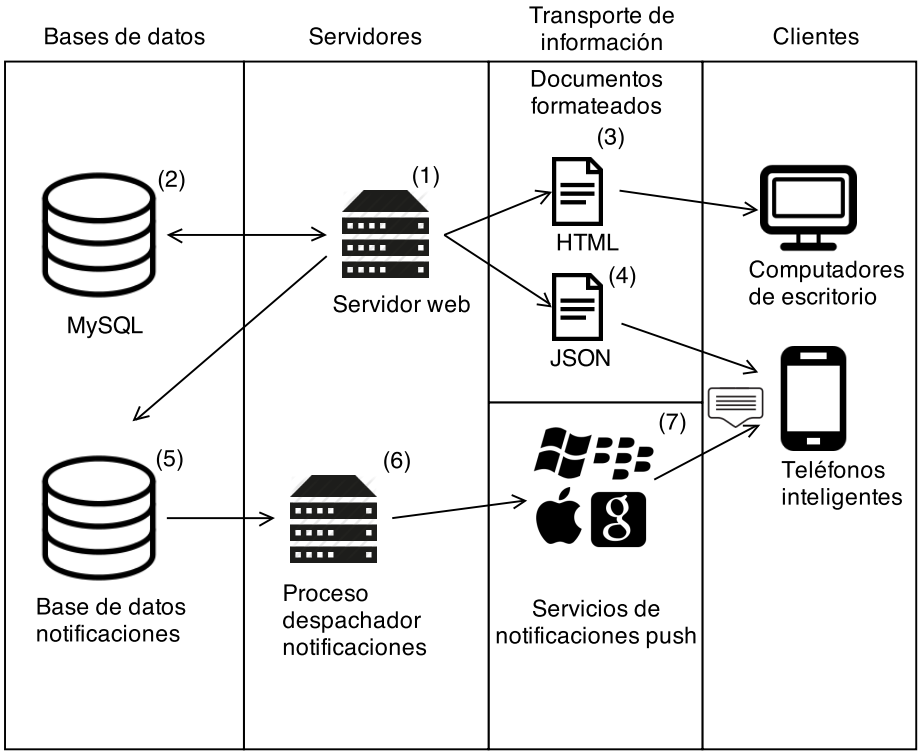

Figura 4.1: Accesos a servicios de U-Cursos por móvil

na sobre HTTP y revisando el análisis presentado en la sección 2.2, hemos decidido adoptar una arquitectura *RESTful* para el acceso de los contenidos por medio de APIs. REST nos ofrece un acceso simple a los recursos utilizando el protocolo HTTP, que ya es utilizado por la plataforma. Además, el esquema de nombres de recursos ya ha sido diseñado e incorporado al sistema actual. Con respecto a las semánticas de interacción, se utilizará Lo-REST ya que actualmente el servicio actualmente utiliza principalmente solo los verbos *POST* y *GET*. Como se mencionó anteriormente, el formato de los contenidos intercambiados en los requerimientos de API serán en formato JSON.

4.3.1. Autenticación

Para la autenticación de usuarios, nos integraremos con el servicio U-Pasaporte. Como vimos en la sección 2.1.5, este servicio es provisto por la Universidad de Chile para identificar a alumnos, académicos y funcionarios. Dado que este sistema es utilizado actualmente por U-Cursos para la autenticación de sus usuarios, por ende es innecesario utilizar un sistema de autenticación paralelo.

4.3.2. Recursos

Para acceder a los canales y servicios de U-Cursos a través de una API REST es necesario definir un esquema de URLs (Universal Resource Locators) consistente para identificar los distintos recursos disponibles. Actualmente U-Cursos posee un esquema de URLs para acceder a los servicios por medio de un navegador web. La Figura 4.2 muestra una URL para el servicio de datos del curso para Trabajo de Título (CC69F). Esta compuesto por *la url del servidor seguro de U-Cursos* (1), la facultad (2), el año del curso (3), el semestre (4), el código del curso (5), la sección del curso (6) y el servicio que se accede (7).

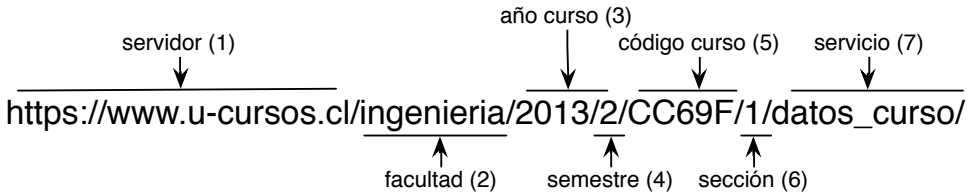

Figura 4.2: Estructura de una URL de U-Cursos

Como es posible notar, la URL es bastante explicita y completa respecto al recurso accedido. Para no inducir cambios innecesarios, para el acceso a recursos a través de la API, mantendremos la estructura actual. Para diferenciar si se accede al recurso como parte de la aplicación web o la API pública, se agrega el prefijo **api/0/** a la ruta del recurso. El número al final del prefijo tiene como objetivo indicar la versión de la API. Dado lo anterior, la URL para el servicio de los Datos de Curso para CC69F queda como en la Figura 4.3, donde se agrega el prefijo correspondiente (8). Para el caso de las APIs, la URL base de un servicio desplegará las acciones disponibles para el servicio, luego es necesario escoger la acción deseada desde el índice y utilizarla de acuerdo a la documentación (Ver Figura 4.4).

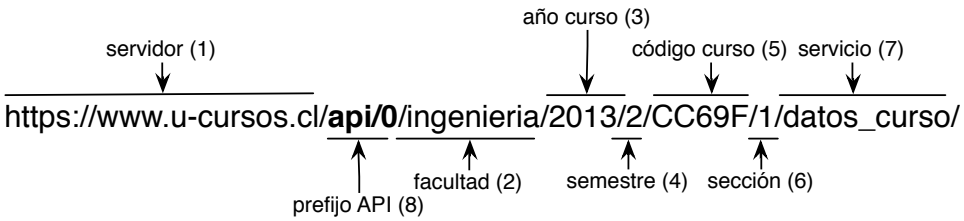

Figura 4.3: Estructura de una URL de la api de U-Cursos

4.4. Notificaciones

Un aspecto importante a considerar en la aplicación móvil de U-Cursos es su conectividad permanente a internet, esto permite informar al usuario de eventos importantes en tiempo real. Para ello los proveedores de sistemas operativos móviles modernos ofrecen servicios de notificaciones push. Estos servicios consisten en una arquitectura distribuida (Ver Figura 4.5) en el que participa el servidor de la aplicación (1), un sistema de despacho que se encarga de enrutar y transportar las notificaciones de forma coherente (2) y los dispositivos móviles que reciben las notificaciones (3). El servidor de la aplicación es administrado por el desarrollador

Figura 4.4: Directorio de servicios para Datos de Curso

de la aplicación móvil y el sistema mediador es provisto por los desarrolladores del sistema operativo del dispositivo. Actualmente Android [4], Blackberry[5], iOS [6] y Windows Phone[7] proveen servicios de notificación push y utilizan todos una arquitectura muy similar.

Gracias a que los servicios de entrega de notificaciones ya están provistos, en nuestro desarrollo debemos enfocarnos en el servidor que genera los mensajes para las aplicaciones móviles. Dado esto, hemos definido una arquitectura para el despacho de notificaciones desde los servidores de U-Cursos. En la Figura 4.6 mostramos los nuestros componentes que se encargarán de la generación y despacho de los mensajes para los dispositivos. Proponemos utilizar servidor de base de datos *key/value* donde se van almacenando las nuevas notificaciones, luego las éstas son consumidas por un proceso dedicado que despacha a los servicios de notificación correspondientes.

Para poder despachar los mensajes a los dispositivos correctos y para evitar abuso del sistema, es necesario poseer un *device token*, el cual debe ser obtenido desde el dispositivo móvil y comunicado al servidor de generación de notificaciones. El servidor mantiene el *device token* asociado al usuario dueño del dispositivo. De este modo, el token sirve de identificador único del destinatario. Dado lo anterior, habilitaremos una URL de registro de los *device tokens* de los dispositivos que requieran notificaciones. Esta URL será parte de las APIs oficiales y será administrada por el servicio *Mis Canales* que es el responsable de la administración de las notificaciones.

[4]http://developer.android.com/google/gcm/gcm.html
[5]http://developer.blackberry.com/bbos/java/documentation/push_service_overview.html
[6]https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html#//apple_ref/doc/uid/TP40008194-CH100-SW9
[7]http://msdn.microsoft.com/en-us/library/hh221549.aspx

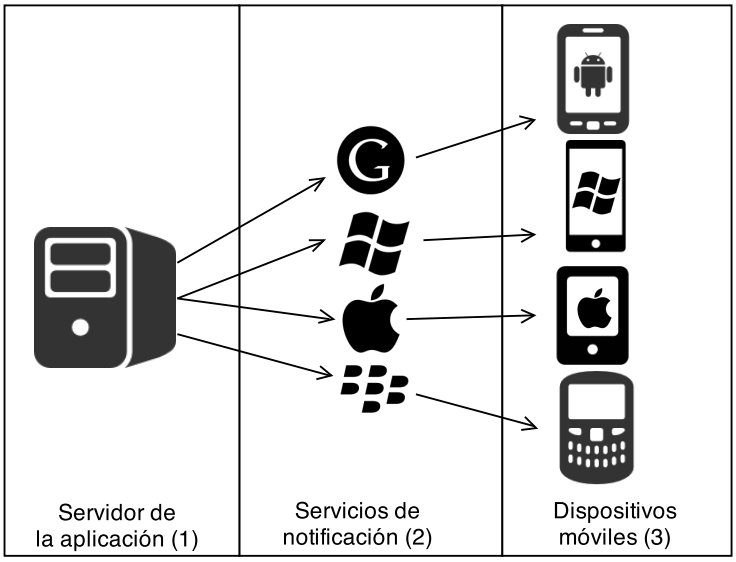

Figura 4.5: Arquitectura general de los servicios de notificaciones push

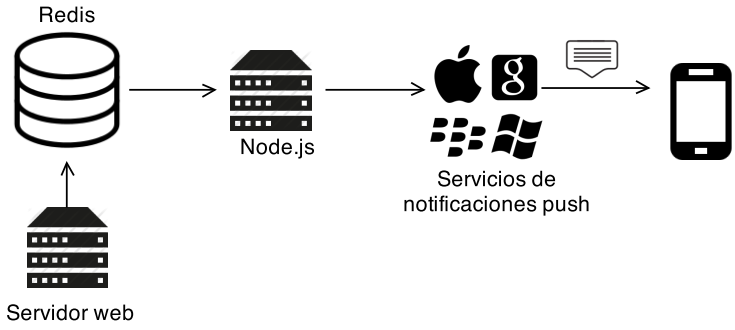

Figura 4.6: Arquitectura para el despacho de mensajes desde la plataforma U-Cursos

4.5. Aplicación móvil

Como vimos en la sección 2.3, existen varias alternativas para mostrar contenido en terminales móviles:

Aplicación web móvil: Se desarrollan vistas web compatibles con dispositivos con pantallas pequeñas utilizando HTML y CSS. Esta alternativa funciona para cualquier aplicación, pero no se aprovechan todos los recursos del móvil y el despliegue de información es menos eficiente que con vistas nativas.

Aplicación nativa: Implementación de una aplicación utilizando los kits de desarrollo na-

tivos. Las aplicaciones son mucho más rápidas y pueden usar todas las características del dispositivo móvil. Lamentablemente requiere mayor trabajo de desarrollo ya que se debe implementar de forma paralela en cada sistema operativo.

Aplicación híbrida: Permiten desarrollar aplicaciones nativas utilizando vistas en HTML. Esto agiliza el desarrollo al producir aplicaciones para cada teléfono desde un mismo código base. Lamentablemente el despliegue de información no es tan rápido como en aplicaciones nativas a causa del uso de HTML incrustado en la aplicación. Además, la integración con todos los dispositivos no suele ser tan directa como se espera, obligando introducir condiciones para solucionar problemas de compatibilidad o incluso a generar ramas paralelas.

Dado que requerimos enviar notificaciones push a los clientes móviles, hemos descartado una aplicación completamente basada en la web. Por otro lado, dado que las aplicaciones híbridas son menos eficientes y no ofrecen el *look&feel* del dispositivo, por lo que hemos preferido implementar la aplicación de manera nativa. Para efectos de la presente memoria, implementaremos una aplicación para dispositivos iOS. Esto es motivado porque ADI ha asignado recursos propios para la implementación en Android, y dado que según nuestro estudio del acceso desde móviles (ver Sección 3.2), iOS ocupa el segundo lugar en utilización.

4.5.1. Funcionalidades

La aplicación iOS a desarrollar consiste en un cliente del servicio U-Cursos, que además de mostrar el contenido relevante para el alumno según su demanda, enviará notificaciones cuando existan eventos de interés para el usuario.

Las notificaciones en la aplicación iOS implementarán utilizando el servicio *Apple Push Notification Service (APNS)*. Como lo mencionamos en la Sección 4.4, Apple provee un servicio que se preocupa de despachar correctamente los mensajes a los dispositivos que le corresponden. Luego, es responsabilidad del desarrollador el implementar el servidor generador de notificaciones y también el manejo de ellas en el dispositivo móvil. Dado esto, cuando la aplicación se encuentre en primer plano (foreground), mostraremos una ventana de dialogo con la descripción de la notificación y las opciones de ver o cancelar. Cuando lleguen nuevas notificaciones y la aplicación se encuentra en segundo plano (background) el sistema operativo se encargará de mostrar el mensaje al usuario en el centro de notificaciones de iOS. Las vistas propuestas se encuentran en la Figura 4.7.

Además, para evitar una sobrecarga de mensajes sin relevancia para el usuario, se le permitirá activar las notificaciones de los servicios que generan actualizaciones. Como se ve en la Figura 4.8, se le presentará una lista agrupada por curso o comunidad, donde se despliegan los servicios. Al hacer click sobre el servicio se activará o desactivara el servicio de notificación para ese canal en particular.

Adicionalmente los usuarios podrán navegar y revisar los servicios de sus cursos del semestre actual. A los cursos actuales se puede acceder a través del botón *mostrar menú* que se muestra en la esquina superior derecha de cualquier vista. En la Figura 4.9 se muestra como

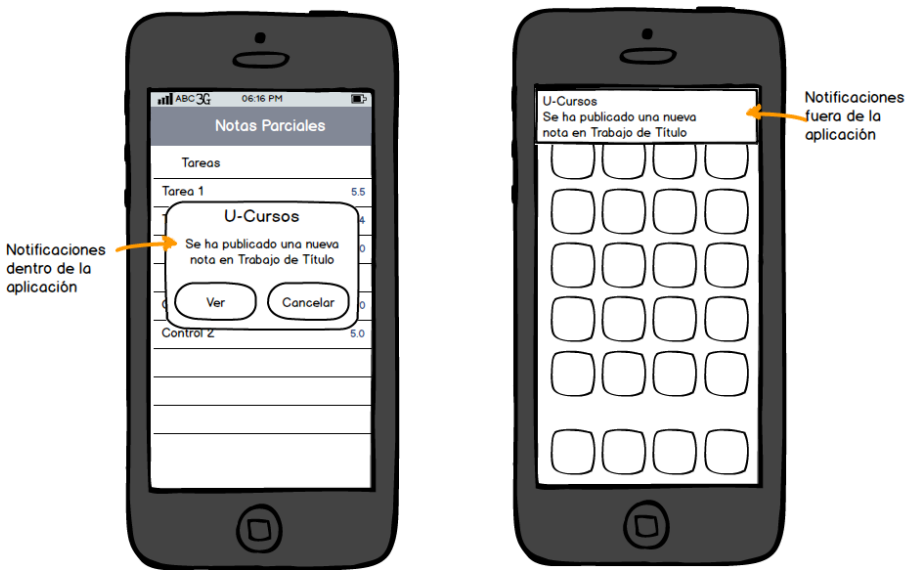

Figura 4.7: Mock de la presentación de notificaciones al usuario

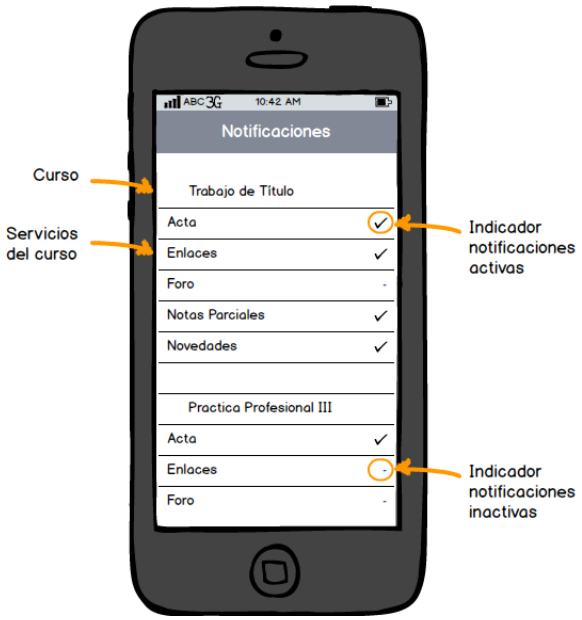

Figura 4.8: Mock de la activación de notificaciones

se despliega el menú de cursos. Luego a partir de ellos se listan los servicios disponibles.

Con respecto con la presentación de servicios de U-Cursos, hemos decidido implementarlos progresivamente desde HTML con soporte para móviles a vistas nativas. Esto es debido a que al existir una gran cantidad de servicios de U-Cursos, el proceso de implementación completo sería muy lento y conllevaría a lanzar una aplicación con menos servicios que la plataforma web actual o retrasaría el lanzamiento de la aplicación. Para esto, proponemos que los servicios que aún no se encuentren implementados nativamente, sean desplegados como contenido HTML en un elemento UIWebView[8] dentro de la aplicación como en la

[8]Elemento de la interfaz gráfica de iOS para desplegar contenido web dentro de una aplicación

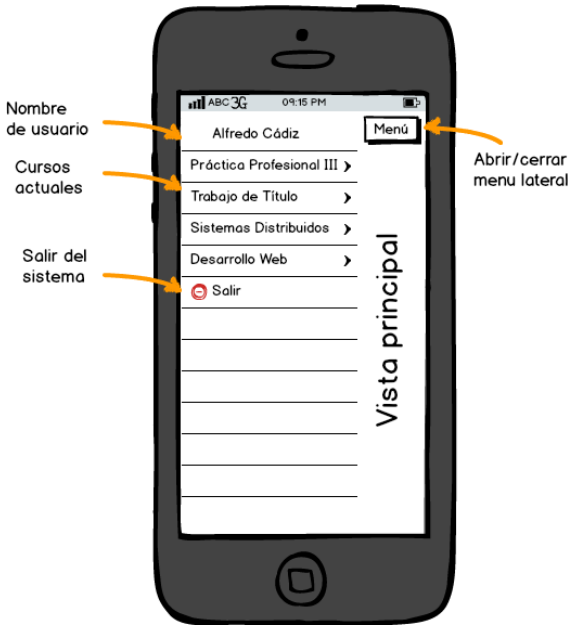

Figura 4.9: Mock del menú lateral

Figura 4.10. Luego, a medida que se van implementando nuevas funcionalidades nativas se liberan versiones actualizadas de la aplicación con ellas.

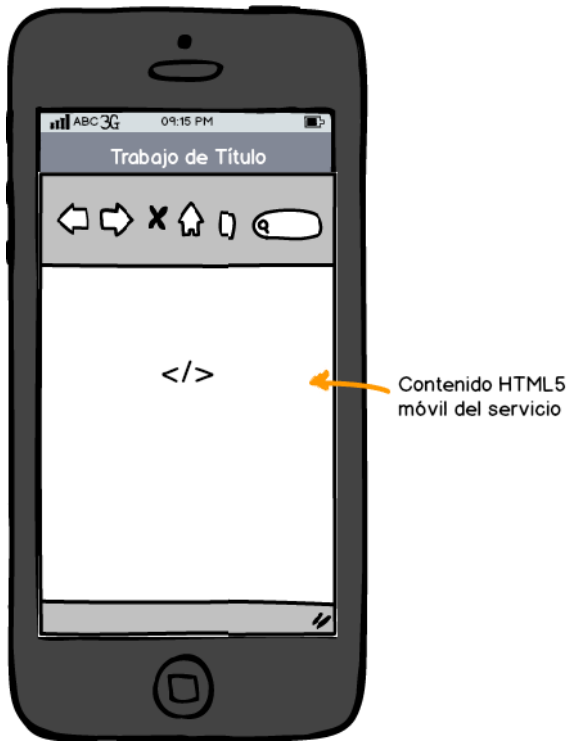

Figura 4.10: Mock de la visualización de vistas web dentro de la aplicación

En el caso de existir una vista nativa de un servicio, se procederá a mostrarla con los datos obtenidos a través de accesos a la API de U-Cursos. Por ejemplo, para el servicio de *Notas Parciales* se propone presentar 2 niveles (ver Figura 4.11). Donde el primero es el resumen de

notas del alumno en una lista y que al seleccionar una de las celdas, se acceder al detalle de ésta. En el detalle se mostrará información como los comentarios generales y personalizados de la nota, el desglose de las distintas partes de la nota y estadísticas como las presentes en la plataforma web.

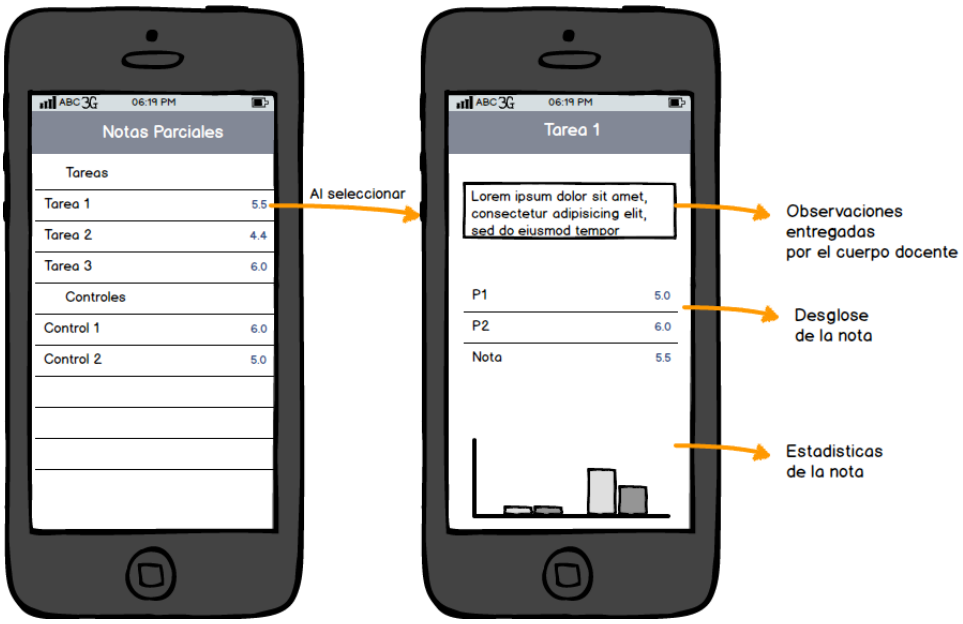

Figura 4.11: Menú de la funcionalidad de notas parciales

Dados los requerimientos recuperados del análisis de la plataforma y el diseño propuesto, en el capítulo siguiente detallaremos la implementación de las APIs y la aplicación móvil.

Capítulo 5

Implementación

Dadas las decisiones y diseño propuestos en el Capítulo 4 describiremos en este capítulo cómo se desarrollaron los distintos aspectos necesarios para implementar el cliente móvil de U-Cursos en la plataforma iOS.

5.1. Acceso a APIs

Las APIs de U-Cursos nos permiten acceder de manera fácil a los datos necesarios para implementar cualquier tipo de aplicación externa, ya sean clientes o procesos automáticos. Esto gracias a que se podrá consultar por los datos a través de los recursos publicados en su API REST. Además, la comunicación de los datos será a través de la notación JSON, que es soportada por la mayoría de los lenguajes de programación modernos.

Las APIs de U-Cursos han sido implementadas principalmente por los ingenieros de ADI. Esto es porque es necesario acceder y modificar código fuente crucial para el servicio y se requiere de conocimientos avanzados en la plataforma para poder desarrollar sobre ella.

Las APIs fueron implementadas en el lenguaje PHP y conviven con la actual servicio web y bases de datos de U-Cursos, además utiliza los roles y permisos ya existentes. Dado que también utiliza el sistema de autenticación U-Pasaporte, el acceso a las vistas web y a las APIs es equivalente en la forma de autenticar y en los roles y permisos aplicados.

Como mencionamos en la Sección 4.3.2, los recursos se encuentran publicados a través de URLs en paralelo a las de la plataforma actual, agregando el prefijo **/api/0** a las rutas actuales. De esta forma el sistema es consistente con la funcionalidad actual.

Por el lado del cliente iOS, hemos implementado bibliotecas de acceso tanto para la autenticación con U-Pasaporte como con el acceso a los recursos a través de la API de U-Cursos. En la Figura 5.1 se muestra como el cliente interactúa con las bibliotecas para acceder a los recursos. En estricto rigor, el cliente solo interactúa con *U-Api*, el cual contiene los métodos que el desarrollador puede utilizar para acceder a los datos de U-Cursos. El flujo de los

requerimientos sigue el indicado en la Figura 2.4.

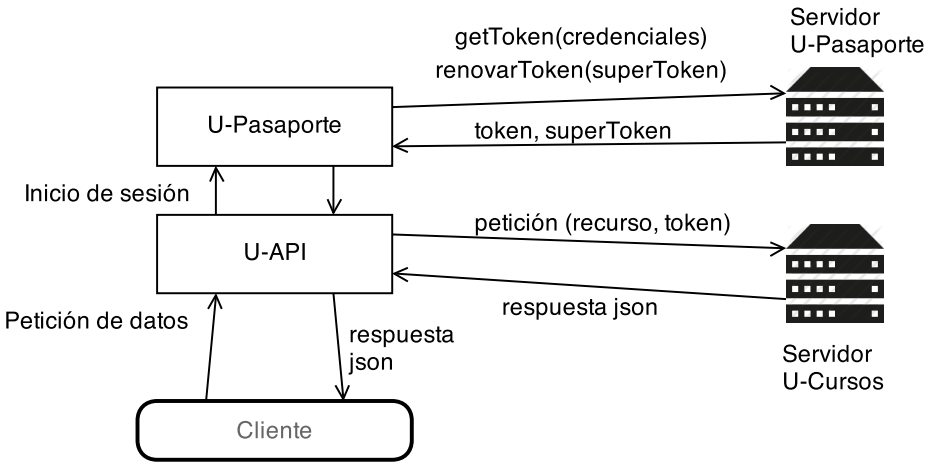

Figura 5.1: Interacción entre el cliente y las bibliotecas

Al recibir requerimiento, U-API revisa si el usuario ha iniciado su sesión, si es así, envía el requerimiento al servidor de U-Cursos utilizado el *token* de acceso previamente adquirido. El servidor entonces retorna la respuesta y U-API la redirecciona a la aplicación. En caso que el usuario no haya iniciado sesión, U-Api utiliza la biblioteca *U-Pasaporte* para requerir el inicio de ésta. U-Pasaporte despliega al usuario una vista de ingreso de credenciales (Figura 5.2), la cuales se envían por canal seguro al servidor de U-Pasaporte. Si las credenciales son correctas se retorna un *token* para ejecutar requerimientos y un *superToken* para renovaciones posteriores.

Adicionalmente, los *tokens* tienen un tiempo de expiración, lo que imposibilita reutilizarlos por mucho tiempo. Cuando un token se invalida, el servidor de U-Cursos rechaza la petición con el código HTTP *403 - Not Authorized*. Ante este escenario, U-Api automáticamente intenta renovar el token, en caso de error se le pedirá al usuario reiniciar su sesión.

Las bibliotecas U-Api y U-Pasaporte se han implementado utilizando el modelo de programación por bloques de Objective-C en lugar del modelo de *callbacks*. Esto es porque el uso de bloques agiliza el desarrollo y permite tener mayor control de las respuestas a los requerimientos asíncronos. Dado esto, el acceso a los recursos de U-Cursos se realiza mediante los métodos *getPath:success:failure:* y *postPath:withParameters:success:failure:* que ejecutan requerimientos GET y POST respectivamente. Ambos métodos se preocupan de realizar la consulta al servidor y de administrar los tokens de U-Pasaporte, minimizando los casos de excepción para el desarrollador. Además los parámetros *success:* y *failure:* en ambos casos indican si el requerimiento fue procesado con éxito o si han surgido errores. El detalle de los métodos públicos de las bibliotecas U-Api y U-Pasaporte se encuentran en el Apéndice B.

Un uso básico de la U-Api se muestra en el Listing 5.1. En la línea 1 se recibe la instancia de la biblioteca a utilizar. En la línea 2 se invoca el método para revisar si la API esta

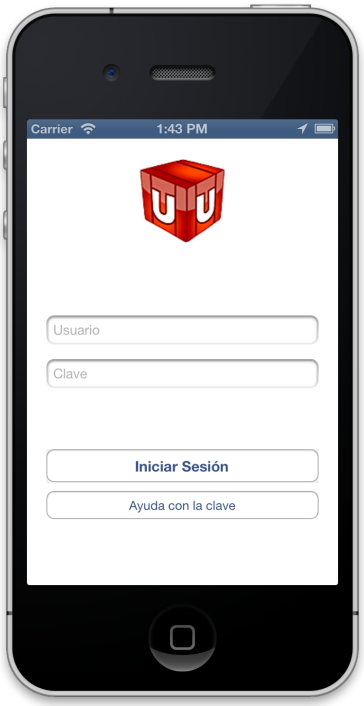

Figura 5.2: Vista de inicio de sesión de U-Pasaporte

preparada para ejecutar. Al ser exitoso este llamado, se realiza el obtención de datos utilizando *getPath:success:failure:* en la línea 3. Entre las líneas 4 y 9 se implementa el bloque para el caso de éxito y entre 10 y 11 para el caso de error.

Listing 5.1: Uso de U-Api

```
1  UApi *uapi = [UApi sharedUApiWithAppToken:@"" withAppSecret:@""];
2  [uapi checkAccountIsActiveWithSuccess:^{
3      [uapi getPath:@"/usuario/_/mis_canales/suscripciones"
4            success:^(NSObject *json) {
5                [self updateData:(NSArray *)json];
6      }
7            failure:^(UApiError error) {
8                NSLog(@"Error receiving data");
9      }];
10  } failure:^(NSError *error) {
11        NSLog(@"Problem loading the api");
12  }];
```

Para la implementación de las bibliotecas U-Api y U-Pasaporte se utilizaron proyectos de terceros para agilizar el desarollo:

UICKeyChainStore [1] Es un envoltorio para facilitar el acceso al módulo Keychain de almacenaje seguro de los dispositivos iOS.

Base64 [2] Es una categoría sobre NSData que permite agregar soporte para codificar y

[1] https://github.com/kishikawakatsumi/UICKeyChainStore
[2] https://github.com/nicklockwood/Base64

decodificar datos en formato base 64.

AFNetworking [3] Es una biblioteca que mejora las interfaces nativas de conexión utilizando el protocolo HTTP.

5.2. Notificaciones

Como mencionamos en la Sección 4.4, la implementación de notificaciones tiene 3 componentes: el servidor de la aplicación, quien genera las notificaciones, el servicio de notificaciones que se encarga de despachar el mensaje al dispositivo, y finalmente el dispositivo móvil quien debe recibir y desplegar la notificación correctamente.

En el caso de iOS, Apple ofrece el servicio *Apple Push Notification Service (APNS)* para despachar los mensajes a los dispositivos y prevenir mal uso de ellos. Por lo tanto, nosotros debemos preocuparnos de proveer el servidor generador de notificaciones y del manejo de éstas en dispositivos móviles. La Figura 5.3 muestra esquema para aplicaciones basadas en iOS. En ella, tanto el *Provider* como el *Client App* son nuestro servidor de notificaciones y aplicación cliente de U-Cursos respectivamente.

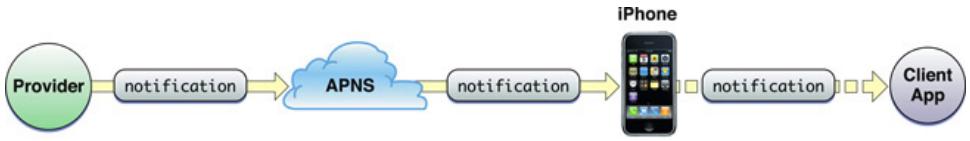

Figura 5.3: Entrega de un mensaje Push en sistemas iOS. Fuente: Apple iOS Developer Library

Para el servidor hemos implementado la arquitectura descrita en la Sección 4.4. Esta arquitectura incluye una base de datos del tipo *key/value*, que para nuestro caso será Redis y un proceso que procesa las notificaciones pendientes, implementado en Node.js.

El flujo de un envío de notificación se inicia al generarse un nuevo contenido desde algún servicio de U-Cursos, entonces el servidor web inserta la nueva notificación en la base de datos Redis. Para procesar los mensajes mantenemos una aplicación implementada en Node.js que consume las notificaciones en Redis. Luego, por cada notificación consumida se realiza consulta a la base de datos MySQL de U-Cursos por los usuarios y sus dispositivos subscritos al canal asociado a la notificación. Finalmente por cada usuario, el proceso Node.js despacha el mensaje a los servicios de notificaciones apropiados según el dispositivo del usuario. Para el caso del servicio de notificaciones push de iOS, utilizamos el módulo *node-apn*[4] para facilitar el envío de los mensajes a los servidores de Apple.

Como mencionamos anteriormente, para que el flujo descrito arriba funcione, es necesario poseer el *device token* de cada destinatario. Para ello debemos implementar un flujo de petición del token y de envío a los servidores de ADI, para su posterior uso para enviar

[3] `http://afnetworking.com/`
[4] `https://github.com/argon/node-apn`

notificaciones. El flujo de registro comienza cuando la aplicación se inicia por el usuario, entonces ella se registra en el *Apple Push Notification Service*. Este servicio provee el *device token* que identifica la aplicación y el dispositivo donde reside. Una vez obtenido el *device token*, la aplicación lo envía al servidor de notificaciones para que sea asociado al usuario registrado en el dispositivo. Para esto, se ha habilitado el servicio de administración de las notificaciones push. Este servicio provee de las URLs necesarios para registrar o borrar los tokens, además permite listar los canales a los que puede registrarse el usuario y escoger las notificaciones que serán enviadas. El detalle de los métodos se encuentra en la Tabla 5.1. El servicio de registro de dispositivos push forma parte de la API de U-Cursos y depende del servicio *Mis Canales*. En este caso la funcionalidad de registro de un *device token* es exclusivo para la API porque por el servicio web no tiene sentido su implementación.

Ruta	Descripción
`api/0/usuario/_/mis_canales/`	Directorio de servicios para notificaciones push.
`api/0/usuario/_/mis_canales/registrar`	Registro de nuevos *tokens* de notificaciones.
`api/0/usuario/_/mis_canales/desregistrar`	Eliminar registro de un *token* de notificaciones.
`api/0/usuario/_/mis_canales/suscripciones`	Lista los canales disponibles para enviar notificaciones al usuario.
`api/0/usuario/_/mis_canales/suscribir`	Permite activar o desactivar las notificaciones para un canal dado

Tabla 5.1: Resumen URLs para administrar notificaciones desde dispositivos móviles

La implementación de las notificaciones por el lado del servidor de U-Cursos fue realizada por ADI. Esto es porque se deben modificar funcionalidades cruciales de la plataforma y requiere de la experiencia de sus ingenieros para evitar errores. La elección de Redis y Node.js para el manejo de las notificaciones obedece a la necesidad de separar su procesamiento de las tareas principales de la plataforma y de aprovechar que tecnologías permiten implementar rápida y eficientemente un esquema *publish/subscribe*. Además, el sistema propuesto es suficientemente genérico para dar soporte para los servicios de notificación más importantes. De este modo, al integrar nuevas plataformas para el servicio de notificaciones, se puede mantener la arquitectura actual y solo se deben agregar el soporte para ellas en el proceso Node.js.

5.3. Aplicación móvil

La aplicación móvil de U-Cursos consiste en un cliente de la API REST de la plataforma. La aplicación que despliega información relativa a los cursos del usuario registrado de manera acorde a las capacidades del dispositivo. Además permite recibir notificaciones push sobre novedades relevantes para el usuario y podrá desplegar rápidamente la vista relacionada a la notificación.

Como mencionamos en la Sección 4.5, para efectos de la presente memoria implementamos un cliente nativo basado en el sistema operativo iOS. Esto es debido a que la implementación para dispositivos Android se asignó ingenieros de ADI. De este modo se generarán versiones móviles de U-Cursos en los dos sistemas operativos móviles más utilizados actualmente.

La aplicación móvil utiliza la biblioteca U-Api, presentada en la Sección 5.1, para acceder a los contenidos de U-Cursos. Además se utilizaron las siguientes bibliotecas externas:

FMDB [5] Biblioteca para facilitar el acceso nativo a las bases de datos SQLite3 de la aplicación.

UITableViewCell+ProgressIndicator [6] Categoria que agrega a las celdas de las tablas nativas la posibilidad de mostrar un indicador de progreso.

SWRevealViewController [7] Biblioteca que permite mostrar menús laterales y manipularlos con sus propios controladores.

UIAlertView-Blocks [8] Biblioteca que permite desplegar ventanas de alerta de manera más simple a utilizando bloques en lugar de *callbacks*.

MBProgressHUD [9] Permite desplegar icono animado de progreso en celdas de tablas nativas cuando se debe esperar la respuesta a algún requerimiento. Por ejemplo al registrar canales para recibir notificaciones push.

SDWebImage [10] Es un cache transparente para imágenes. De este modo se minimizan las descargas para imágenes recurrentes.

La aplicación iOS tiene 3 aspectos principales: configuración y recepción de notificaciones, servicios nativos, servicios web.

5.3.1. Manejo de notificaciones

Como mencionamos anteriormente, la aplicación móvil de U-Cursos tendrá soporte para notificaciones push. Estas notificaciones son recibidas asíncronamente por el dispositivo. Esto quiere decir que pueden llegar mientras la aplicación se encuentra en primer plano, o también, cuando se encuentra en segundo plano o cerrada. La Figura 5.4 muestra como se presentan las notificaciones: cuando la aplicación se encuentra en primer plano se muestra una alerta para que el usuario pueda escoger ver la nueva información o cancelar. En caso de que la notificación se reciba cuando la aplicación se encuentra en segundo plano, el sistema operativo se encarga de mostrar el mensaje.

[5] `https://github.com/ccgus/fmdb`

[6] `https://github.com/c99koder/lastfm-iphone/blob/master/Classes/UITableViewCell%2BProgressIndicator.m`

[7] `https://github.com/John-Lluch/SWRevealViewController`

[8] `https://github.com/jivadevoe/UIAlertView-Blocks`

[9] `https://github.com/jdg/MBProgressHUD`

[10] `https://github.com/rs/SDWebImage`

Figura 5.4: Notificaciones en primer plano (izquierda), notificaciones recibidas con la aplicación en segundo plano (centro) y el centro de notificaciones con las últimas notificaciones recibidas (derecha).

Además debemos preocuparnos de darle al usuario la flexibilidad de escoger los canales de los que recibirá notificaciones. Esto es porque cada nuevo evento es potencialmente enviado al dispositivo y si un canal genera demasiadas notificaciones generará problemas de usabilidad y de eficiencia. Por la parte usabilidad, los usuarios verán gran cantidad de mensajes desde un solo canal y generará dificultades para revisar otros eventos. Por la eficiencia, producto el alto volumen de mensajes puede afectar la duración de la batería del dispositivo por el uso de la red y de pantalla. Actualmente existen canales que generan gran cantidad de mensajes, como los foros de facultades pueden generar más de 1000 mensajes durante un día.

En la aplicación iOS de U-Cursos presentamos la vista de la Figura 5.5, que le muestra al usuario sus cursos y servicios actuales y le permite escoger cuales le generarán notificaciones. Además en la misma vista el usuario puede activar o desactivar todas las notificaciones.

5.3.2. Vistas nativas

Los servicios nativos son aquellos servicios que han sido implementados con elementos gráficos del sistema operativo móvil en la aplicación móvil. Estos servicios acceden a los datos de U-Cursos a través de las APIs establecidas y despliegan la información utilizando las interfaces del sistema.

La estrategia a seguir para la implementación de U-Cursos en dispositivos móviles es la proveer las vistas esenciales para la configuración de las notificaciones push y la navegación de los cursos y sus servicios. Además para cada servicio se proveerá ya sea su implementación nativa o un acceso a la vista web asociada. De este modo proveeremos todos los servicios disponibles desde la primera versión de la aplicación y las vistas nativas serán implementadas a medida que son necesarias por la aplicación.

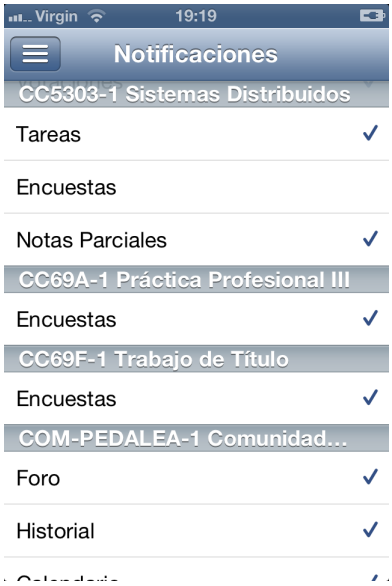

Figura 5.5: Configuración de las notificaciones

Una primera vista nativa implementada es la de la configuración de notificaciones, ya mencionada en la sección anterior (Ver Figura 5.5). Además en la Figura 5.6 se muestran las vistas nativas de cursos actuales y los servicios de cada uno. En ella el usuario puede revisar sus cursos actuales (izquierda) para luego ver cuales servicios se encuentran disponibles (derecha). Al seleccionar alguno de los servicios se accede a la vista del servicio escogido.

Figura 5.6: Lista de cursos y servicios en aplicación en iOS

Para el caso del servicio de *Notas Parciales* para un alumno de un curso, hemos implementado su versión nativa, como se puede ver en la Figura 5.7. Esto fue hecho en 2 vistas: la primera muestra la nota final de todas las evaluaciones publicadas y la segunda se muestra al seleccionar alguna de las evaluaciones, desplegando la información detallada de ésta.

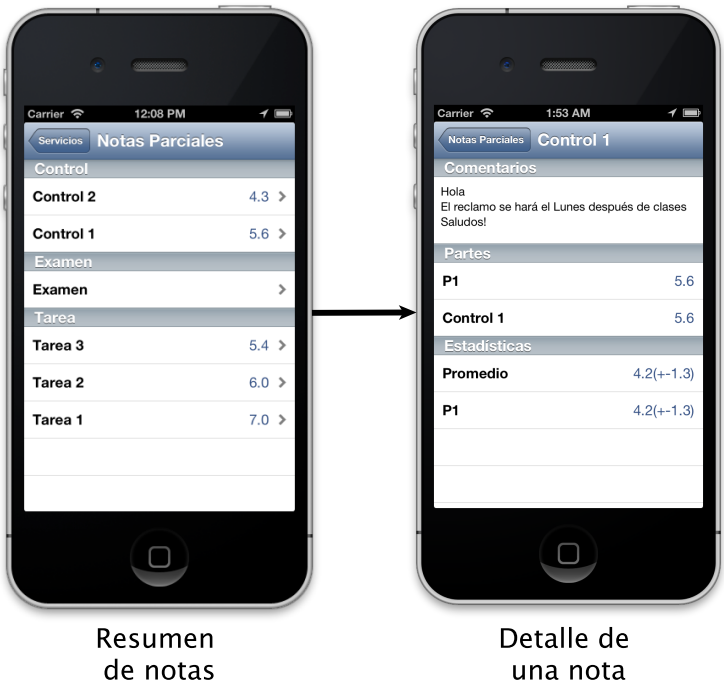

Figura 5.7: Servicio *Notas Parciales* en vistas nativas iOS

5.3.3. Vistas web

Para los casos en que los servicios no hayan sido desarrollados con componentes nativos de iOS, hemos definido un acceso híbrido a través de vistas web de los servicios. Esto se ha implementado a través de una vista nativa con un componente *UIWebView* para mostrar contenido web incrustado en la aplicación. En la Figura 5.8 se muestra el despliegue de la información de algunos servicios en las vistas híbridas.

Para optimizar el espacio en pantalla, se han generado vistas web de U-Cursos apropiadas para móviles. Al pedirle al servidor contenido web para móviles, este envía solamente el contenido principal del servicio, descartando los menues lateral y superior. Además el contenido transmitido posee optimizaciones adicionales para mostrarse correctamente en pantallas pequeñas.

Figura 5.8: Servicios de U-Cursos a través de vistas web incrustadas en la aplicación

Capítulo 6

Resultados

Dado el análisis del Capítulo 3, el diseño planteado en el Capitulo 4 y la implementación descrita en el Capítulo 5, hemos logrado implementar una versión funcional de la aplicación móvil de U-Cursos. En este capitulo detallamos los resultados relevantes obtenidos en cada etapa del proceso.

6.1. Análisis

Hemos analizado U-Cursos a través de los datos de los registros de acceso del servidor web de la plataforma. Además hemos estudiado las necesidades de los usuarios a través de la información provista a través de una encuesta en línea realizada en el mismo sistema. Esto nos ha permitido conocer los comportamientos actuales de los usuarios y sus expectativas sobre un cliente móvil de U-Cursos.

Respecto al análisis de registros, hemos analizado 100Gb de registros de accesos a la plataforma del mes de Abril. A partir de estos registros hemos podido identificar los dispositivos utilizados para ingresar al sistema y a qué servicios se utilizaron. Dado esto, hemos podido observar que existe un acceso muy bajo al sistema desde dispositivos móviles: 8 % sumados iOS y Android (Figura 3.14). Hemos constatado que esto no concuerda con los altos índices de penetración de la internet móvil y los teléfonos inteligentes en Chile. Adicionalmente hemos analizado los acceso de los servicios de U-Cursos a través de los dispositivos más importantes actualmente en la industria de sistemas operativos móviles: iOS y Android. Hemos corroborado que con el estado del sistema actual los servicios más populares son *foros*, *material docente*, *notas parciales*, *material alumnos* y las *novedades* (Ver Figura 3.1).

Por parte de la encuesta a los usuarios, hemos publicado en U-Cursos una encuesta en línea durante la primera semana de Septiembre de 2013. En ella se consultó a los usuarios sobre la relevancia de servicios actuales y el interés de servicios nuevos que puedan ofrecerse en una versión móvil de U-Cursos. Durante los 7 días de la encuesta, se recibieron más de 4000 (10 % de los usuarios activos) respuestas. En ellas se muestra un interés muy claro por los servicios de *notas parciales*, *foros*, *novedades*, *material docente* y *tareas*. Por lo que, por un

lado, hemos podido corroborar el interés de los usuarios en obtener servicios en sus teléfonos inteligentes, y por otro lado las expectativas de los usuarios están alineadas con el uso actual de la plataforma desde móviles.

Como resultado de esta parte, hemos generado resultados que muestran el estado actual del sistema con respecto al acceso móvil. Además hemos podido obtener retroalimentación de los usuarios con sus opiniones respecto al cliente móvil.

6.2. Diseño

Hemos recopilado los requisitos del cliente móvil de U-Cursos y hemos detallado las características más relevantes: Notificaciones push, acceso a servicios actuales, eficiencia en móviles. Además hemos indicado que es necesario crear APIs para el acceso de los datos de U-Cursos a través de clientes externos, esto permitirá habilitar la plataforma no solo para software creado dentro de ADI sino que además para dar soporte a colaboradores externos. Para ello se define como requisito que se autentique con el sistema U-Pasaporte, además se pide extensibilidad de la API y el uso de formatos estándares.

Dado lo anterior, hemos propuesto una arquitectura para dar soporte a las APIs de acceso a los contenidos de U-Cursos y también para las notificaciones push (Figura 4.1). Además hemos definido que el esquema de URLs a utiliza será basado en el actual esquema del servicio web, que es consistente y permite identificar de forma clara los recursos.

Finalmente, dados los requerimientos del sistema, hemos seleccionado la estrategia de implementación de la aplicación. Será una aplicación nativa iOS que utiliza las APIs de U-Cursos para implementar funcionalidades escenciales, como la configuración de notificaciones push y listado de cursos. Además se implementarán nativamente los servicios más importantes para que su acceso sea eficiente. Para el caso de los servicios no implementados en la aplicación, también se podrá acceder a ellos a través de una vista híbrida desplegando contenido web en formato de móviles.

De este modo el diseño de la arquitectura del servicio es compatible no solo con un cliente iOS sino que con cualquier cliente que desee utilizar los contenidos de U-Cursos. Además para el caso de las notificaciones, dado que el esquema propuesto es suficientemente genérico, se podrá extender para dar soporte a otros sistemas de avisos en tiempo real. Finalmente en el caso de la aplicación iOS, ésta brinda funcionalidades nativas como las notificaciones push y despliegue de algunos servicios. Pero también ofrece acceso rápido y optimizado a cualquier otro servicio de la plataforma a través de sus vistas híbridas.

6.3. Implementación

Dado el diseño propuesto, hemos implementado el cliente nativo de U-Cursos para sistemas iOS. La implementación incluyó el desarrollo de dos bibliotecas para facilitar la interacción

con los sistemas de ADI: *U-Pasaporte* y *U-Api*. U-Pasaporte permite encapsular el proceso de obtención y renovación de *tokens* para el uso de las APIs de U-Cursos. U-Api facilita el acceso a las APIs de U-Cursos a través de una interfaz simplificada que aliviana al desarrollar de tareas relacionadas a la comunicación con el servidor.

Con respecto a las notificaciones, se han agregado a las APIs los recursos necesarios para poder recibir registro de *device tokens* desde los distintos dispositivos que utilicen la funcionalidad. Además por lado del despacho, se ha puesto en marcha por parte de ADI del servidor de datos *key/value* Redis y de las instancias de Node.js para procesar notificaciones.

Finalmente se ha implementado la aplicación en iOS que despliega la información de U-Cursos utilizando la API de U-Cursos, en el caso de las vistas nativas o despliega contenido HTML optimizado para móviles en sus vistas híbridas. Además registra las notificaciones push con el servidor de la aplicación y despliega las notificaciones que recibe.

El código fuente tanto de las bibliotecas U-Pasaporte y U-Api, como el del cliente se encuentra versionado en servidores Git de ADI.

Capítulo 7

Conclusión

El trabajo realizado en esta memoria nos ha permitido analizar el estado actual de la plataforma U-Cursos actual respecto al acceso a ella por dispositivos móviles. Además hemos propuesto e implementado una arquitectura para el soporte de acceso a los contenidos de la plataforma a través de APIs públicas y el manejo de notificaciones push. Finalmente hemos implementado un cliente en iOS utilizando las APIs y el sistema de notificaciones.

Durante la fase de análisis hemos podido recopilar información desde los servidores del sistema a través de sus registros de accesos y desde los usuarios a través de encuestas. Gracias a esto hemos constatado una acceso muy limitado por parte de dispositivos móviles a la plataforma y una gran expectativa de los usuarios en un cliente móvil.

La propuesta de diseño para soportar APIs de los servicios permite dar acceso a los contenidos de U-Cursos por parte de cualquier cliente externo de forma eficiente y confiable. Además la arquitectura de las notificaciones da soporte para la mayoría de los sistemas presentes actualmente, permitiendo que el sistema sea extensible a otros servicios de notificaciones cuando sea necesario.

La implementación de un cliente iOS funcional muestra que el diseño propuesto es factible y permite un acceso más eficiente a los contenidos de U-Cursos a través de dispositivos móviles.

El trabajo desarrollado en esta memoria sienta las bases para lograr un cliente con calidad de producción que pueda ser utilizado por los usuarios de la plataforma. Esperamos que la aplicación, cuando sea lanzada, permita mejorar la calidad del acceso móvil a la plataforma e incrementar su cuota de uso, también alivianando la carga del servicio en general al disminuir la incertidumbre de nuevos contenidos en canales relevantes para los usuarios y con ello disminuir accesos para verificar actualizaciones.

7.1. Trabajo futuro

Si bien se ha completado una versión inicial del cliente iOS de U-Cursos, la biblioteca de acceso a U-Pasaporte, la biblioteca de acceso a las APIs de U-Cursos y la implantación de la arquitectura para el soporte de notificaciones push, quedan tareas pendientes para poder hacer este sistema accesible por todos los usuarios.

Es necesario trabajar junto con ADI en la mejora de la calidad de los datos publicados a través de las APIs. Es necesario probar y uniformizar lo mejor posible los datos generados. Además es necesario extender la API a todos los servicios que aún no la tienen.

Adicionalmente, debemos coordinar el diseño gráfico de las vistas nativas iOS con las vistas web para que la navegación por los recursos se sienta integrada y no parezcan sistemas distintos.

Al igual que las APIs, debemos mejorar la calidad del cliente móvil para que sea bien recibido por los usuarios. Para esto debemos iniciar rondas de *alpha testing* y *beta testing* para corregir errores y mejorar características de la aplicación.

Junto con las sesiones de *beta testing* y cuando a aplicación se libere para todos los usuarios, es necesario monitorear los accesos a través de la aplicación móvil para medir el impacto del cliente en los accesos al servicio tanto en el volumen total como en los acceso desde móviles.

Capítulo 8

Bibliografía

[1] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple object access protocol (soap) 1.1, 2000.

[2] Canalys. Smart phones overtake client PCs in 2011. Technical report, Canalys, 2012. Available at `http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011`.

[3] Cristián Céspedes, Manuel Ortega, , Julio Salas, and Javier Villanueva. U-cursos 3.0: Un enfoque hacia los ambientes comunitarios y las facilidades de uso. In *XXI Congreso Chileno de Educación en Ingeniería*. Santiago de Chile, 2007.

[4] Cristián Céspedes and Julio Salas. U-cursos: Una plataforma de apoyo a la docencia presencial. In *XX Congreso Chileno de Educación en Ingeniería*. Viña del Mar, 2006.

[5] Andre Charland and Brian LeRoux. Mobile application development: Web vs. native. *Queue*, 9(4):20:20–20:28, April 2011.

[6] K. G. Coffman and A. M. Odlyzko. Internet growth: Is there a "moore's law" for data traffic? In *HANDBOOK OF MASSIVE DATA SETS*, pages 47–93. Kluwer Academic, 2001.

[7] Subsecretaría de Telecomunicaciones de Chile. Radiografía de servicios de internet fija y móvil, 2012. `http://www.subtel.gob.cl/images/stories/apoyo_articulos/notas_prensa/radiografia_internet_06septiembre2012.pdf`.

[8] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000. AAI9980887.

[9] Mark H. Goadrich and Michael P. Rogers. Smart smartphone development: ios versus android. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, SIGCSE '11, pages 607–612, New York, NY, USA, 2011. ACM.

[10] Dick Hardt. The oauth 2.0 authorization framework. 2012.

[11] Tim O'Reilly. What is web 2.0: Design patterns and business models for the next generation of software. MPRA Paper 4578, University Library of Munich, Germany, March 2007.

[12] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. "big" web services: making the right architectural decision. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 805–814, New York, NY, USA, 2008. ACM.

[13] David Recordon and Drummond Reed. Openid 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management*, pages 11–16. ACM, 2006.

[14] Paul Sagan and Tom Leighton. The internet & the future of news. *Daedalus*, 139(2):119–125, Apr 2010.

[15] S.J. Vaughan-Nichols. Will html 5 restandardize the web? *Computer*, 43(4):13–15, 2010.

Apéndice A

Datos encuesta U-Cursos

Servicios actuales	Servicios nuevos móvil	Servicios nuevos general
Mi Horario (92)	Push granular (20)	Chat (39)
Correos (74)	Previsualizar Material (11)	Biblioteca (7)
Notificaciones (70)	Buscar Sala (10)	Multiplataforma (6)
Material Docente (58)	Simplicidad (9)	Tablets (6)
Calendario (50)	Multiplataforma (6)	Integración Facebook (4)
Notas Parciales (46)	Marcar Asistencia (5)	Mapa Facultades (3)
Foro (20)	Buscar Personas (4)	Estado salas (2)
Dropbox (14)	Offline (4)	Horario Profesor (2)
Notas Guía (11)	Reservar Laboratorio (4)	Leer después (2)
Tareas (7)	Línea de tiempo (4)	Reclamos DIM (2)
Novedades (7)	Solicitud Certificados (4)	Virtual Classroom (2)
Acceso a cursos pasados (5)	Widget Android (4)	Banco de Apunte (1)
Favoritos (3)	Calculadora Notas (4)	Beneficios Salud (1)
Actas (2)	Coordinador (3)	Buscar baños (1)
BIA (2)	Google Drive (3)	Integrar Evernote (1)
Encuestas (2)	Colas de impresión (2)	Feedback anónimo cursos (1)
Enlaces (2)	Correo Institucional (2)	Integrar Foursquare (1)
Histórico de notas (2)	Inscripción académica (2)	Formar grupos (1)
Sesión persistente (2)	Botón +1 y -1 (1)	Comprar horario compañeros (1)
Administración Cursos (1)	Catálogo de cursos (1)	Redes sociales (1)
Afiches (1)	CSS móvil (1)	Reporte error inscripción (1)
	Foro docente (1)	
	Encontrar oficinas docentes (1)	
	Encontrar sala (1)	
	Encontrar SEMDA (1)	
	Encontrar casinos (1)	
	Accesos rápidos (1)	

Tabla A.1: Resumen conceptos en caja de texto libre en Encuesta U-Cursos

Apéndice B

Documentación bibliotecas

Describiremos en este capítulo los métodos públicos de las bibliotecas U-Api y U-Pasaporte. Normalmente un desarrollador solo deberá utilizar los métodos de U-Api ya que este último utiliza los de U-Pasaporte internamente.

B.1. U-Api

U-Api esta implementado como una variante del patrón singleton. Para utilizar la biblioteca es necesario obtener la instancia común a través del método de clase *sharedUApiWithAppToken:withAppSecret:*. A continuación es necesario revisar si la Api esta preparada para ser utilizada, para ello se invoca el método *checkAccountIsActiveWithSuccess:failure:*. Una vez que se haya ejecutado del bloque *success* de este método es posible utilizar la biblioteca de la API.

B.1.1. sharedUApiWithAppToken:withAppSecret:

```
+ (UApi *)sharedUApiWithAppToken:(NSString *)appToken
        withAppSecret:(NSString *) appSecret
```

Obtiene una instancia de U-Api para el *AppToken* y *AppSecret* entregados.
Parámetros:

appToken: App token entregado por U-Cursos.

appSecret: App secret entregado por U-Cursos.

Retorna:

UApi * La instancia de U-Api a utilizar.

B.1.2. checkAccountIsActiveWithSuccess:failure:

```
- (void) checkAccountIsActiveWithSuccess:(void(^)()) success
        failure:(void (^)(NSError *error))failure
```

Revisa si la API esta en condiciones para ser utilizada. Revisa si las credenciales se encuentran presentes, sino, mostrará al usuario la ventana de ingreso para iniciar la sesión. Si logra iniciarse ejecutará el bloque *success*, si no hay forma de iniciarse ejecutará el bloque *failure*.
Parámetros:

success: Bloque a invocar cuando la Api esté lista para ser usada.

failure: Bloque a invocar cuando la Api haya fallado en sus intentos por inicializarse.

B.1.3. registerKey:success:failure:

```
- (void) registerKey:(NSString *)key
        success: (void (^)())success
        failure: (void (^)(NSError *error)) failure;
```

Registra una llave de notificaciones push en U-Cursos.
Parámetros:

key: Llave de notificaciones push del dispositivo.

success: Bloque a ejecutarse si el registro es exitoso.

failure: Bloque a ejecutarse si el registro ha fallado.

B.1.4. unRegisterKey:success:failure:

```
- (void) unRegisterKey:(NSString *)key
        success: (void (^)())success
        failure: (void (^)(NSError *error)) failure
```

Remueve una llave de notificaciones push en U-Cursos.
Parámetros:

key: Llave de notificaciones push del dispositivo.

success: Bloque a ejecutarse si la operación fue exitosa.

failure: Bloque a ejecutarse si la operación ha fallado.

B.1.5. getPath:success:failure:

```
- (void) getPath:(NSString *)path
        success:(void (^)(NSObject *json))success
        failure:(void (^)(UApiError error)) failure
```

Consulta por un recurso en U-Cursos en el *path* indicado. Si es exitoso, se invocará el bloque *success*, entregando el resultado de la consulta en formato JSON. En caso contrario se invocará *failure* indicando la razón.
Parámetros:

path: Ruta del recurso de U-Cursos a consultar.

success: Bloque a ejecutarse si la operación fue exitosa.

failure: Bloque a ejecutarse si la operación ha fallado.

B.1.6. postPath:withParameters:success:failure:

```
- (void) postPath: (NSString *) path
        withParameters: (NSDictionary *) parameters
        success:(void (^)(NSObject *json))success
        failure:(void (^)(UApiError error)) failure
```

Realiza un requerimiento de cambio de estado en un recurso en U-Cursos en el *path* indicado con los datos entregados en *parameters*. Si es exitoso, se invocará el bloque *success*, entregando el resultado de la operación en formato JSON. En caso contrario se invocará *failure* indicando la razón.
Parámetros:

path: Ruta del recurso de la U-Cursos a modificar.

parameters: Diccionario conteniendo los parámetros a entregar.

success: Bloque a ejecutarse si la operación fue exitosa.

failure: Bloque a ejecutarse si la operación ha fallado.

B.2. U-Pasaporte

La biblioteca U-Pasaporte tiene como fin separar el inicio de sesión de las funcionalidades de U-Api.

Contiene dos métodos públicos: *requestSuperTokenForAppKey:withAppSecret:success:failure:* para iniciar una sesión y *requestTokenWithSuperToken:success:failure:* para renovar un token expirado.

B.2.1. requestSuperTokenForAppKey:withAppSecret:success:failure:

```
- (void) requestSuperTokenForAppKey:(NSString *) appKey
        withAppSecret:(NSString *) appSecret
        success:(void (^) (NSString *token, NSString *superToken))success
        failure:(void (^) (NSError *error)) failure
```

Inicia sesión con U-Pasaporte. Se encarga de obtener las credenciales del usuario y contactarse con el servidor de autenticación. Si la operación es exitosa se invoca el bloque *success* con el tokens necesarios para acceder a las APIs.
Parámetros:

appKey: App key de la aplicación.

withAppSecret: App secret de la aplicación.

success: Bloque a ejecutarse si la operación fue exitosa. Incluye el *token* y *superToken* para poder utilizar la API de U-Cursos.

failure: Bloque a ejecutarse si la operación ha fallado.

B.2.2. requestTokenWithSuperToken:success:failure:

```
- (void) requestTokenWithSuperToken:(NSString *) superToken
   success: (void (^)(NSString *newToken, NSString *superToken))success
   failure: (void (^)(UPasaporteError error))failure;
```

Renueva un token expirado utilizando el *superToken* actual. Si la operación es exitosa se invoca el bloque *success* con los nuevos token y superToken.
Parámetros:

superToken: Último superToken entregado en el inicio de sesión o en la renovación anterior.

success: Bloque a ejecutarse si la operación fue exitosa. Incluye los nuevo *token* y *superToken* para poder utilizar la API de U-Cursos.

failure: Bloque a ejecutarse si la operación ha fallado.