Juan Manuel Mougan
Nro. reg: 02-040029-5
DNI: 31659212
(011) 156-176-8124

Trabajo Final - Ingeniería en Informática

Diciembre de 2015

RESUMEN

El presente trabajo final plantea un prototipo totalmente funcional de una plataforma de mensajería móvil, para mejorar la comunicación entre profesores y alumnos de la facultad de ingeniería. Su motivación es la frecuente falta de un medio de comunicación eficiente entre profesores y alumnos, frente a diversos imprevistos que los primeros puedan sufrir, y la pérdida de tiempo y dificultades que ello acarrea en los segundos. Se concluirá el presente trabajo con una demostración de la plataforma mediante el uso de sendas aplicaciones de ejemplo, y sus ventajas frente a otros canales de comunicación, como el e-mail y los grupos de mensajería.

OBJETIVO

Desarrollar una plataforma de mensajería, basada en notificaciones de tipo push, para facilitar la comunicación entre profesores y alumnos, evitando los inconvenientes que presentan otro tipo de canales más tradicionales.

CONCEPTOS

¿Qué son las notificaciones push?

Las notificaciones push permiten recibir avisos de nuevos mensajes o eventos, aún cuando los usuarios no están trabajando de manera activa con la aplicación. En los dispositivos Android, cuando el aparato recibe una notificación, se puede mostrar un mensaje (y agregar un icono personalizado) en la barra de estado del sistema operativo, que al ser presionado pueda realizar una acción predeterminada. Éstas pueden ser enviadas a todos los usuarios de la aplicación, o bien a un determinado subconjunto de ellos. Ésta última forma de envío se denomina multicast.

El servidor que se demuestra hace uso de multicast para enviar notificaciones a un determinado subconjunto de usuarios de la aplicación, que son definidos a través de listas de

subscripción. De esta manera, realizando una única petición HTTP, se puede realizar el envío de manera sencilla y transparente a la aplicación cliente.

Para implementar el envío de estas notificaciones, se utilizó un servicio externo: Google Cloud Messaging (GCM). Las ventajas de esta plataforma, y las razones que llevaron a su elección se detallan más adelante.

ALCANCE

El alcance del proyecto consistió en la implementación de los siguientes componentes del sistema:

- Una aplicación Android nativa, que permita a los alumnos registrarse en el servidor web central (backend), y recibir aquellos mensajes de su interés
- Un backend web, que permita gestionar las notificaciones que los profesores le envían a los alumnos, y diseñar reglas de envío según varios criterios
- Una segunda aplicación Android nativa, que permita a los profesores enviar notificaciones a aquellos alumnos que estén suscritos a listas de su incumbencia

Adicionalmente, se realizó este breve informe, y un tutorial para facilitar la instalación de la aplicación web.

ASPECTOS METODOLÓGICOS

Kanban como metodología de trabajo

Para orientar el proceso de desarrollo de la plataforma, se utilizó una metodología ágil de trabajo, Kanban en particular. Obviamente, teniendo en cuenta que el proyecto fue encarado sólo por una persona, no tiene sentido hablar de “equipo”. Sin embargo, puede considerarse al tutor como el Product Owner del proyecto (utilizando la terminología de Scrum), debido a que tiene una visión general de lo que se desea construir.

Para explayarnos un poco más sobre cuál es su rol, según Scrum la persona más importante del equipo es el Product Owner. Éste trabaja con los stakeholders, representando sus intereses frente al resto del equipo, y a su vez es la primer persona en hacerse responsable por el éxito del equipo. También, provee una dirección de trabajo y objetivos al equipo, y prioriza las tareas a realizar

Algunas características de Kanban:

- No hay división en sprints, el trabajo es continuo

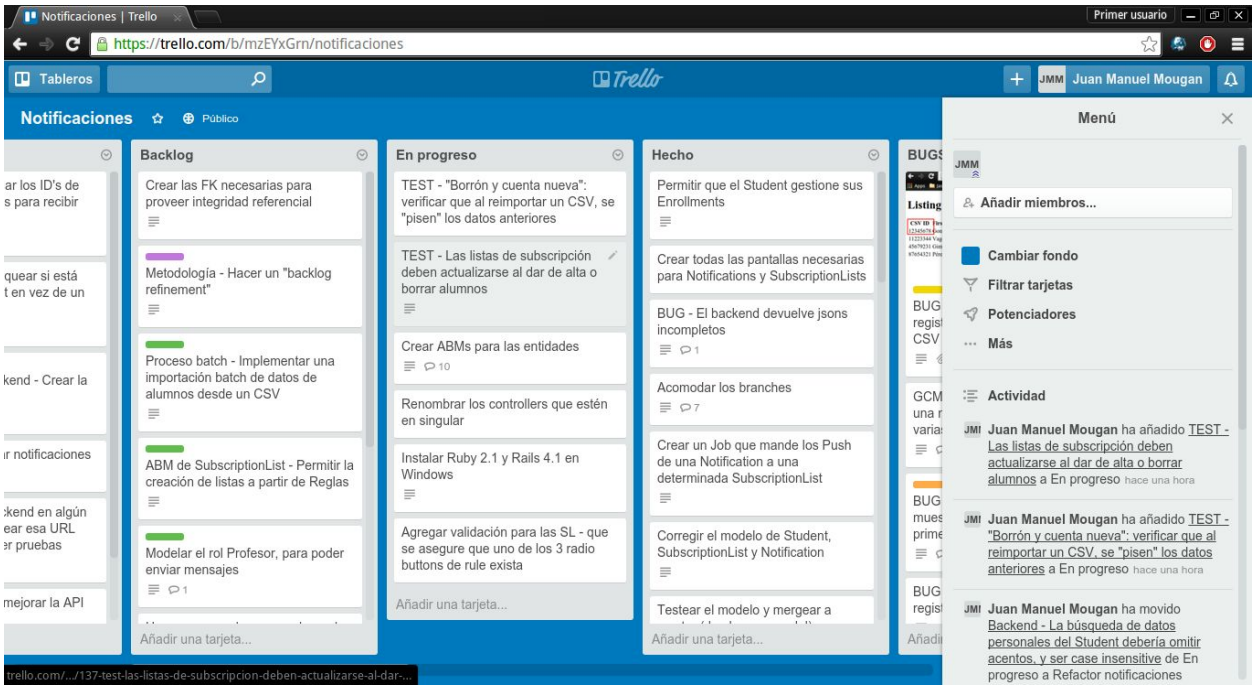
- Las tareas y user stories se muestran en una pizarra (“track board”)
- Dicha pizarra puede tener cualquier cantidad deseada de columnas, que muestran el estado en que se encuentra cada tarea. Por ejemplo: Backlog, planeada, en desarrollo, para testear, hecha
- Se pueden agregar “cortes” transversales a las columnas, por ejemplo, para separar tareas por su prioridad
- Se define un límite de tareas por columna

Trello como herramienta de gestión de proyectos

Para aplicar la metodología elegida de manera organizada y eficiente, se utilizó la herramienta Trello como principal software de gestión del proyecto. Ésta es una aplicación web que permite aplicar Kanban de manera sencilla, exponiendo una pizarra que contiene listas de tareas en forma de columnas, que a su vez contienen tarjetas (“cards”), que representan user stories o tareas del proyecto.

Además de su evidente sencillez, la principal ventaja de esta herramienta es que opera bajo el modelo *freemium*, por lo que se utilizó la versión gratuita para gestionar las tareas. Un punto en contra es que no permite un seguimiento de las horas utilizadas, aunque permita establecer un “vencimiento” para cada tarjeta. De todas maneras, como no existió una restricción de fecha para el desarrollo, esto no fue un inconveniente para este caso en particular.

Un ejemplo de cómo se utilizó la aplicación puede visualizarse en la siguiente imagen.



La pizarra correspondiente al proyecto puede consultarse en el siguiente enlace: https://trello.com/b/mzEYxGrn/notificaciones

ANÁLISIS FUNCIONAL

¿Qué son user stories?

Las user stories son una manera de expresar requerimientos a muy alto nivel. El texto expresado en ellas cubre el rol de los usuarios en el sistema, la necesidad expresada por ellos, y los beneficios que acarreará el uso del software a desarrollar.

A veces son confundidas con Casos de Uso o Escenarios, pero la principal diferencia es que son mucho más cortas, y no describen las interfaces de la aplicación, ni tampoco flujos ni pasos de la misma.

User stories del proyecto

A continuación, se bosquejan posibles user stories para este proyecto, que serán utilizadas como guía a la hora de implementar el mismo. Las tareas que se crearán están derivadas de éstas. No deben considerarse user stories completas, ya que por ejemplo, les falta el criterio de aceptación. Son solamente reglas generales de desarrollo.

1. Proceso CSV

   Como un usuario de la Secretaría de la facultad, necesito poder procesar un archivo CSV provisto, para ingresar al sistema datos de los Alumnos y sus Inscripciones a Materias.

2. Creación de reglas para envío

   Como un usuario de la Secretaría de la facultad, deseo poder crear reglas de envío de mensajes a los Alumnos, en base a ciertos criterios preestablecidos.

3. Modelar el rol Profesor

   Como un profesor de la facultad, quisiera poder enviar mensajes a aquellas listas de envío de mi interés, utilizando mi propio dispositivo móvil.

4. Modelar el rol Alumno

   Como un alumno de la facultad, deseo poder recibir mensajes de mi interés en mi dispositivo móvil.

Stakeholders del proyecto

Según la Scrum Alliance, los stakeholders son aquellas personas que tienen deseos y expectativas, y son la razón por la cual el equipo está desarrollando software en primer lugar. Para el proyecto que nos concierne, podemos definir tres stakeholders:

- Alumno
- Profesor
- Secretaría

Los primeros están interesados en recibir avisos de su interés por parte de los profesores, los segundos están interesados en enviar dichos avisos de manera sencilla y efectiva, y los terceros desean poder administrar información de interés para la facultad, y eventualmente también poder mandar sus propios mensajes.

ARQUITECTURA GENERAL DE LA SOLUCIÓN

Componentes principales

Existen tres componentes fundamentales en la solución propuesta:

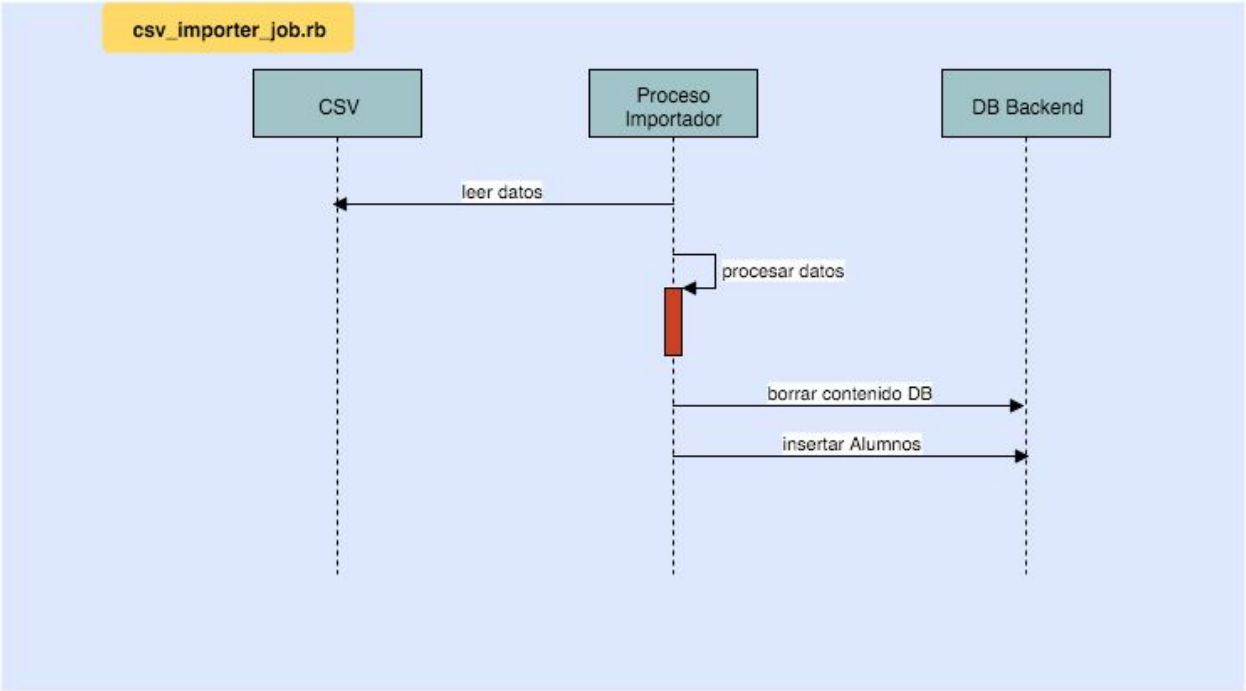
1. Una aplicación Android, denominada “Enviar Notificaciones”
2. Una aplicación web, denominada “Backend”
3. Una segunda aplicación Android, denominada “Notificaciones”

Adicionalmente, la distribución de mensajes a aquellas instancias de la aplicación cliente que estén interesadas, se realiza a través de un cuarto componente: el servicio GCM, provisto de manera gratuita por Google.

Proceso de importación de datos al sistema

Como precondición para poder utilizar el sistema, los datos de carreras, materias y alumnos deben estar precargados en el sistema. Los dos primeros son datos poco cambiantes, por lo que pueden insertarse vía scripts Ruby, o bien código SQL. Para el tercero, se creó un proceso aparte, que lee e interpreta un archivo CSV que contiene un registro por cada cursada de cada alumno de la facultad, e inserta en la base de datos del backend al alumno (Student) con sus respectiva lista de inscripciones a materias (Enrollments).

Una explicación a más bajo nivel de cómo funciona este proceso se puede observar en la imagen que se muestra a continuación:

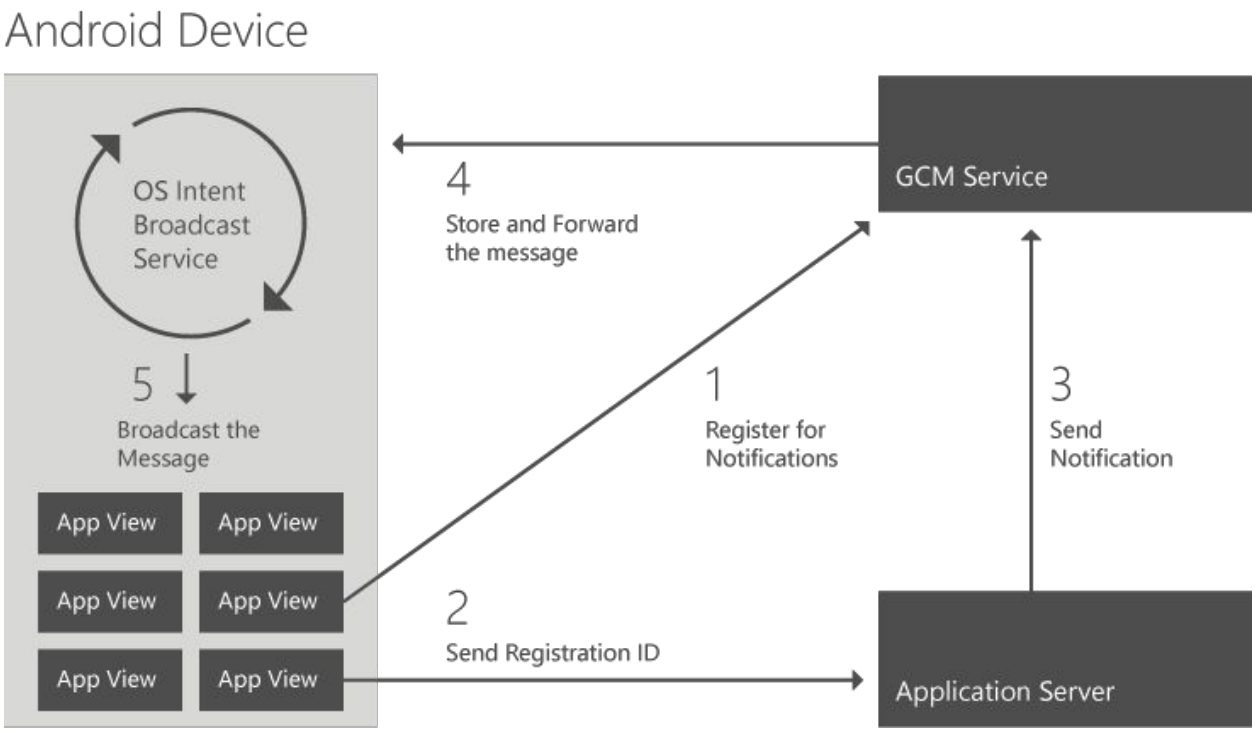


Transporte: Google Cloud Messaging

GCM (Google Cloud Messaging) es un servicio gratuito provisto por Google, que reemplaza a su predecesor actualmente desactualizado, C2DM (Cloud to Device Mobile). Permite enviar información desde un servidor a los dispositivos de los usuarios, y también recibir datos enviados por dichos dispositivos, utilizando una única conexión.

Este servicio se encarga de todo tipo de aspectos relacionados con la mensajería, como ser el encolado y la entrega a los clientes, de manera transparente al servidor y a las aplicaciones cliente.

En el siguiente gráfico se ilustra el proceso de envío de mensajes desde el backend (en este caso, denominado “Application Server”), a las aplicaciones cliente, a través de GCM. Los pasos 1 y 2 se ejecutan únicamente al instalar la aplicación, o bien al actualizar los datos personales del Alumno, mientras que los pasos 3, 4 y 5 se ejecutan al enviar una Notificación, o bien desde el servidor central, o bien desde la aplicación para profesores.



Utilizando GCM

Para poder utilizar GCM, se debe ingresar en https://console.developers.google.com y seguir las instrucciones para poder registrar una nueva aplicación. Luego, puede elegirse entre usar HTTPS o XMPP para enviar mensajes al servicio de Google. Se eligió la primera opción, ya que permite realizar multicast de manera nativa, hasta 1000 destinatarios por vez.

Para facilitar aún más el uso de GCM, se utilizó una gem (es decir, una biblioteca, según la terminología de Ruby) que simplifica enormemente la interacción con esta plataforma. La biblioteca en cuestión es gcm, desarrollada por la empresa alemana SpacialDB, y mantenida de manera open source. Ésta provee una API muy sencilla, que permite construir un Hash conteniendo la información que tendrá el mensaje a enviar, utilizando exclusivamente código Ruby, sin necesidad de por ejemplo construir JSONs a mano.

Servicios web REST

REST define un conjunto de principios de arquitectura que permiten diseñar servicios web que se basan en los recursos de un sistema, incluyendo cómo el estado de dichos recursos es accedido y transferido a través de HTTP por una gran variedad de aplicaciones cliente, escritos en distintos lenguajes para una gran variedad de plataformas distintas.

Si medimos el impacto que ha tenido esta tecnología por la cantidad de servicios web que la utilizan, REST ha emergido en los últimos años como el modelo predominante para escribir

dichos servicios. En verdad, su impacto ha sido tan grande, que ha desplazado a SOAP y su enfoque basado en WSDLs, debido a su mayor sencillez y facilidad de uso.

En su forma más pura, una implementación concreta de un servicio REST presenta las siguientes cuatro características:

- Usa métodos de HTTP de manera explícita
- Las llamadas no tienen estado
- Expone direcciones (URIs) de una manera similar a una estructura de directorios
- Permite transferir datos usando XML, JSON o ambos formatos

En esta implementación, se exponen servicios web REST para permitir a las aplicaciones cliente interactuar con el servidor central, permitiendo registrar al Alumno en el backend, consultar las Listas de Subscripción existentes, y enviar una nueva Notificación a una lista.

La lista de servicios expuestos se detalla a continuación.

- Para las Notificaciones:
    - Un servicio POST que permite crear (y enviar) una nueva Notificación
- Para las Listas de Subscripción:
    - Un servicio GET que permite recuperar todas las listas disponibles
- Para los Alumnos:
    - Un servicio POST que permite registrar un nuevo Alumno. Sin embargo, la aplicación Notificaciones no hace uso de este servicio, sino que sólo intenta actualizar un servicio ya existente
    - Un servicio GET para obtener una lista de alumnos, que opcionalmente recibe como parámetros un nombre, apellido y número de legajo, lo que le permite actuar como filtro
    - Un servicio PUT que permite actualizar un Alumno, recibiendo como parámetro su id. Éste es utilizado por la aplicación de Notificaciones para enviarle al servidor central el id que obtuvo por parte de GCM, permitiendo de esta manera que dicho Alumno pueda recibir las notificaciones pertinentes.

Aplicaciones móviles nativas

- Ventajas
    - La principal ventaja de elegir una aplicación nativa por sobre una híbrida, es la performance que se obtendrá. Los frameworks multiplataforma, en tiempo de ejecución, abren un componente nativo de tipo WebView, que no es otra cosa que un pequeño navegador web, que contendrá a la aplicación que

desarrollemos. Ésta capa extra de software entre nuestra aplicación y el sistema operativo perjudica la usabilidad y genera una degradación en la performance.
    - Otra ventaja importante es que el aspecto de la aplicación (el look and feel) se verá en un dispositivo exactamente de la misma manera que se lo puede ver durante la etapa de diseño. Cuando usamos una herramienta multiplataforma, ésta no hace uso de componentes nativos del sistema operativo, sino que intenta emularlos utilizando técnicas de desarrollo web. Por ende, en aplicaciones que muestren una interfaz gráfica lo suficientemente elaborada, ésta diferencia en el aspecto será apreciable al cambiar de plataforma.
    - Una tercera ventaja de las aplicaciones nativas es que GCM puede utilizarse de manera directa, mientras que si se utiliza un framework multiplataforma, pueden necesitarse plugins externos o bibliotecas no oficiales, que pueden presentar bugs inesperados en su funcionamiento.
- Desventajas frente a aplicaciones híbridas
    - La principal característica que hace favorable el uso de un framework que permita desarrollar aplicaciones móviles híbridas, es el tiempo destinado al desarrollo. En vez de construir una versión para cada sistema operativo que se busque abarcar, simplemente se escribe una única aplicación utilizando herramientas de desarrollo web (es decir, HTML, CSS y JavaScript), y el framework elegido se encarga de permitirnos disponer de varias instancias diferentes de nuestra aplicación, compatibles con cada uno de los sistemas operativos elegidos.
- Conclusión
    - Se decidió usar una aplicación nativa por varias de las ventajas presentadas, agregándosele otro tema importante: Como desarrollador, tengo cierta experiencia trabajando con aplicaciones Android nativas, mientras que sólo tuve una breve capacitación en el framework Ionic (híbrido), por lo que estoy mejor preparado para encarar este desarrollo.

DIAGRAMA DE COMO INTERACTUAN LOS COMPONENTES!

TECNOLOGÍAS UTILIZADAS

Lista de tecnologías utilizadas

- Para las aplicaciones móviles: Android SDK, Java como lenguaje de programación
- Para el backend web: Ruby on Rails, framework web full-stack

Proceso de selección de tecnologías

Como parte del proceso de selección de tecnologías, se investigó respecto a cuáles son los lenguajes de programación más populares de los últimos años, para facilitar las tareas de mantenimiento futuro de la plataforma por parte de otros programadores, y tener a su vez la mayor disponibilidad posible de bibliotecas, herramientas y plugins que puedan ser útiles.

Se utilizaron tres ránkings para considerar esta decisión:

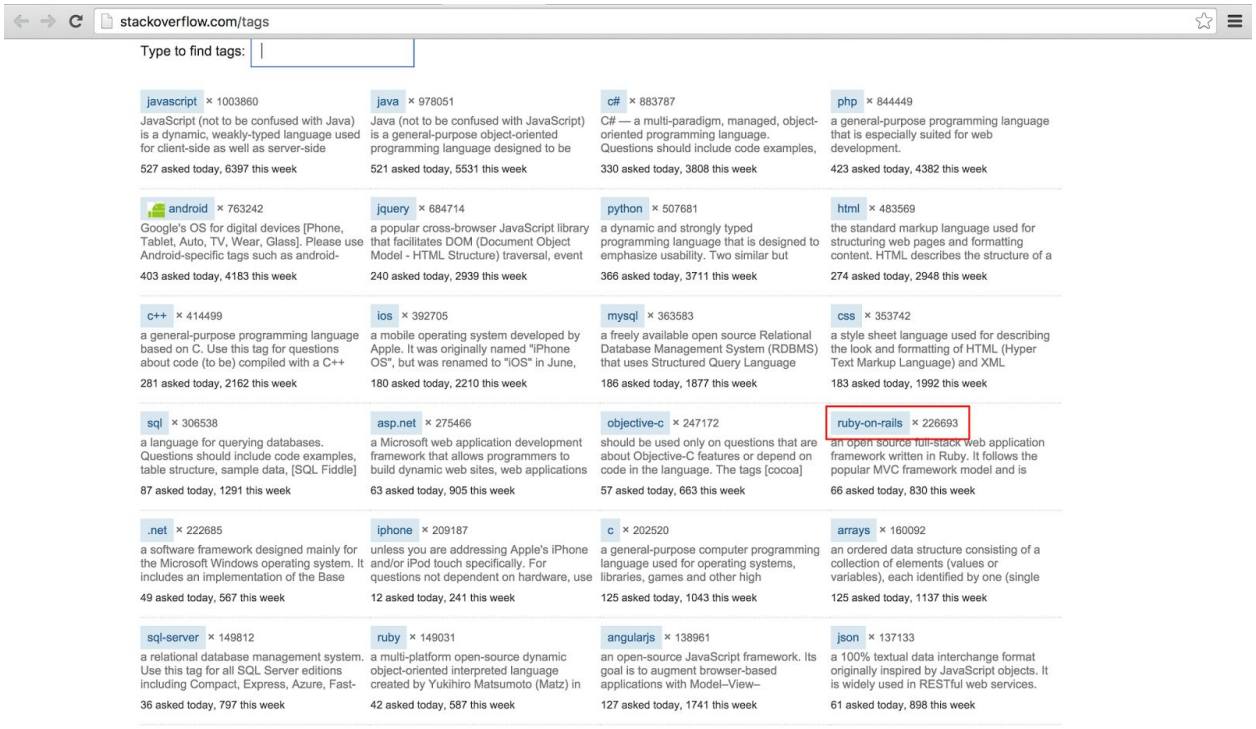
- IEEE Spectrum - The 2015 Top Ten Programming Languages
- The RedMonk Programming Language Rankings: June 2015
- TIOBE Index for November 2015

De los ránkings anteriormente citados, podemos extraer los siguientes datos:

- Java es el lenguaje de programación más popular según dos de ellos (JavaScript lo supera en el índice RedMonk)
- Ruby está en el top 10 en todos los ránkings elegidos (trepando hasta el quinto puesto según RedMonk)

Respecto a los frameworks web, se utilizó una métrica más sencilla, las menciones en etiquetas del popular sitio web StackOverflow, que se puede consultar en el siguiente enlace: http://stackoverflow.com/tags

Podemos observar que Ruby on Rails es el segundo framework web más popular según este indicador, después de ASP.NET, que no es open source, ni permite ser utilizado completamente de manera gratuita. La decisión de utilizar Ruby on Rails es una consecuencia natural de que, a pesar que existan lenguajes de programación más populares que ofrezcan frameworks full-stack, dichos frameworks no son tan utilizados en proyectos como Rails.



Herramientas usadas

Para el desarrollo de la plataforma, se utilizó la siguiente lista de herramientas:

- Trello como herramienta sencilla de gestión del proyecto.
- Android Studio como IDE para desarrollar las aplicaciones Android. Éste es el software para desarrollo sugerido por Google, y está basado en IntelliJ IDEA. Es una herramienta gratuita, e incluye una versión de emulador de Android.
- Genymotion como reemplazo del emulador stock de Android. Funciona mucho más rápido, y permite instalar Google Play Services, para poder recibir las notificaciones push que envía el backend. Es gratuito para uso personal.
- Sublime Text como editor de texto, utilizado para la parte web de la plataforma. Es un software muy popular, y dispone de una gran cantidad de plugins para mejorar su funcionalidad.
- SQLiteStudio como administrador de bases de datos para SQLite 3, la DB por defecto para entornos de desarrollo utilizada por Ruby on Rails. Es gratuita, portable, y open source.
- El código fuente de todas las aplicaciones está disponible en mi cuenta personal de Github, que también es gratuito para proyectos open source. Para realizar el control de versiones se utilizó Git.

CONCLUSIÓN

Ventajas

Si bien no se podrá hacer un análisis exhaustivo de las ventajas provistas por la utilización de la plataforma hasta que ésta se encuentre debidamente instalada en un ambiente productivo, se considera a priori que su utilización acarreará los siguientes beneficios:

- Una comunicación más directa entre profesores y alumnos, debido a la naturaleza de las notificaciones push
- Una adecuada protección de la privacidad del alumno, debido a que los profesores sólo envían mensajes a listas de su interés. Esto evita las cadenas de emails, el intercambio de números de celular, etc. De esta forma, la comunicación es transparente a los datos personales de profesores y alumnos.
- El servicio utilizado para la comunicación (Google Cloud Messaging) garantiza una entrega eventual de los mensajes, y evita que los alumnos tengan que consultar periódicamente sus casillas de email.

Posibles mejoras a futuro

Si bien esta plataforma es totalmente funcional, y puede ser instalada y utilizada en su estado de desarrollo actual, obviamente siempre existe lugar para realizar mejoras y ampliaciones. Por ejemplo, se pueden citar las siguientes:

- Se pueden realizar versiones equivalentes de las aplicaciones móviles para otros sistemas operativos, además de Android. Por ejemplo, según varios ránkings disponibles en la web, el sistema operativo creado por Google es el más popular, seguido de cerca por iOS, con Windows Phone completando el podio. Por lo tanto, una posible alternativa sería replicar la funcionalidad de la aplicación Notificaciones para éstas dos plataformas. De ser requerido por el staff de profesores, se podría hacer lo propio con la aplicación Enviar Notificaciones.
- El proceso de testing de las tres aplicaciones fue totalmente manual, lo que puede llevar a que se filtren ciertos bugs, por no probar todas las combinaciones necesarias luego de agregar nuevas funcionalidades, o de ejecutar tests de regresión luego de corregir errores existentes. Por este motivo, se puede agregar una suite de tests automáticos, y eventualmente, ejecutarlos a través de un servidor de integración continua, para maximizar su eficiencia.