UNIVERSIDAD NACIONAL DEL SUR

TESIS DE LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN

Sistema Automatizado De Divulgación De Información Para Personas Desaparecidas

Alumno: Patricio Emiliano Sartore

Directora: Dr. María Laura Cobo

Departamento de Ciencias e Ingeniería de la Computación

Universidad Nacional del Sur – Bahía Blanca - Argentina

Diciembre del 2016

Índice de Contenido

Introducción

En la actualidad los índices de personas desaparecidas o extraviadas, tanto a nivel mundial como a nivel nacional, son muy elevados y en un alto porcentaje de casos, nunca son encontradas.

En el año 2015, en Argentina se denunciaron un total de 3231 mujeres desaparecidas, la mayoría con edad en el rango de 12 a 18 años. Entre las principales causas de este tipo de desapariciones, pueden mencionarse la trata de personas, la violencia de género y los problemas intrafamiliares.

Si bien se cuenta con muchas organizaciones no gubernamentales encargadas en divulgar la información sobre las personas desaparecidas y así ayudar a su encuentro, en muchos casos, los medios para distribuir la información no son eficientes en cuanto a la velocidad de notificación, como así tampoco para alcanzar a un número considerable de personas con la notificación.

Unos de los mecanismos más utilizados en la actualidad, es la difusión de las mismas por las redes sociales como Facebook, Twitter o Instagram. La problemática que esto conlleva, es la veracidad de la información que se comparte por estas redes ya que no siempre pueden constatarse su origen, pudiendo ser información errónea o falsa.

La distribución actual de notificaciones también adolece de criterio, de forma tal que, la información de estas personas desaparecidas sea dirigida y pensada de manera directa para un usuario en concreto. Por lo general las difusiones son generalizadas, y no se aplican de manera que primero abarque la zona en donde se manifestó el hecho, y luego sea expandida progresivamente hasta abarcar todo el país.

Es por lo mencionado anteriormente, que se propone realizar un sistema cuyo principal objetivo es la automatización de la divulgación de información de las personas que se

ausentan o desaparecen de sus hogares. La funcionalidad introduce la idea de sostener un mecanismo de reportes a los usuarios registrados en el sistema, y así poder generar una divulgación de la información mucho más directa hacia dichos usuarios.

Como objetivo principal, estos reportes deben de ser generados únicamente cuando es de necesidad notificar a los usuarios que, una persona que se encuentra desaparecida pueda llegar a encontrarse en su zona de residencia. Esto se vale de la suposición que dicha persona desaparecida puede ser movilizada involuntariamente desde su lugar de origen hacia el resto del país. De esta forma se evita que diariamente sean enviadas excesivas notificaciones al usuario, y este pierda el interés por ellas.

Estos reportes serán manifestados como notificaciones en sus dispositivos móviles, y así tener un acceso directo a la información de la persona desaparecida en conjunto con fotografías para poder reconocerla en caso de dar con ella.

En el presente trabajo, se detallará la arquitectura general del proyecto en conjunto con cada uno de los componentes que conforman el sistema en su totalidad, proporcionando la información puntual acerca de los lenguajes y tecnologías que se van a utilizar para desarrollar el proyecto. Para justificar dichas elecciones, se contratarán con otras tecnologías y lenguajes, en conjunto con los fundamentos necesarios para llevar a cabo el proceso.

Organización de la Tesis

Este trabajo de encuentra organizado de la siguiente manera:

En el capítulo siguiente, se desarrollará y explicará la arquitectura que adoptará el sistema. Se introducirá brevemente cada uno de los componentes, que serán detallados en los capítulos siguientes.

Para el capítulo subsiguiente, se introducirán los conceptos básicos acerca de los frameworks para el desarrollo web como así el patrón arquitectónico MVC que estos utilizan. Se explicarán y clasificarán las herramientas candidatas para llevar a cabo la implementación del Component Management System (de ahora en adelante CMS), así como cuales fueron los criterios de evaluación para determinar la que mejor se ajuste para su desarrollo.

Posteriormente, se presentará la herramienta Firebase utilizada para el desarrollo de aplicaciones móviles, junto con la descripción de sus componentes y las ventajas de hacer uso de ellos. Se explicará el servicio utilizado para realizar el envío de las notificaciones hacia los usuarios del sistema.

Como siguiente paso, se describirá el sistema de notificaciones. Se desarrollará la explicación del algoritmo empleado para poder llevar acabo el sistema de notificaciones, en conjunto con la estructura de datos que el mismo utiliza y los tiempos de ejecución resultantes. Adicionalmente se presentarán alterativas de estructuras que se contrastarán y evaluarán para alcanzar a la elección de la misma.

Se introducirá, a continuación, un capítulo para describir la noción del protocolo de comunicación SOAP, en conjunto con su funcionamiento y estructura. Se planteará las razones por las cuales fue elegido y cuál es su finalidad de uso dentro del proyecto.

También, se desarrollará brevemente la tecnología WSDL utilizada para la manipulación de los XML's.

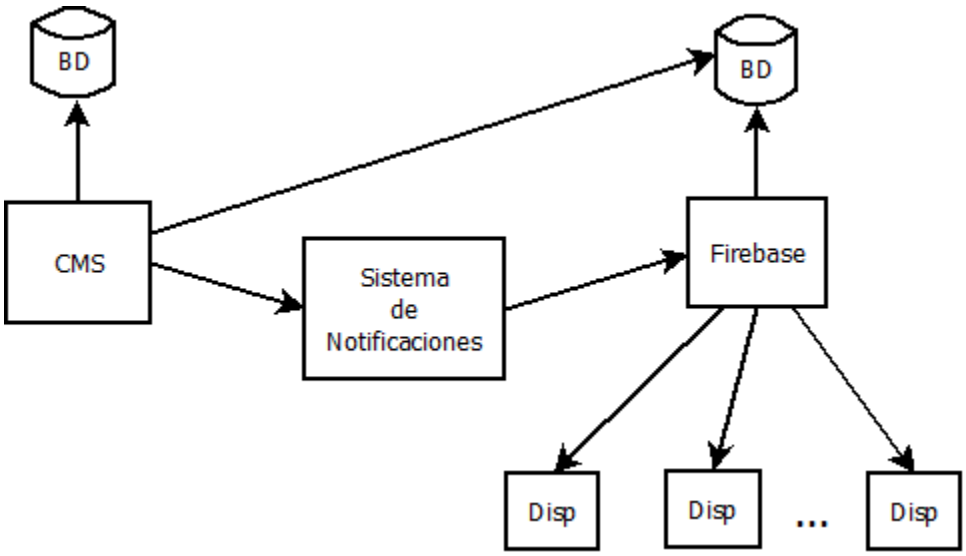
Finalmente, se presentarán las concusiones acerca del proyecto, involucrando todo lo visto a lo largo del desarrollo del informe, como las tecnologías adoptadas, las estructuras necesarias y utilización de sistemas externos.

Capítulo 1 - Arquitectura del Sistema

Se desarrollará y explicará la arquitectura que adoptará el sistema. Se introducirá brevemente cada uno de los componentes, que serán detallados en los capítulos siguientes.

1.1 Esquema de Arquitectura del Sistema

A continuación, se hará presentación de esquema de arquitectura de sistema que se adoptará a lo largo de la implementación final del proyecto.



Como se puede apreciar en la imagen, tanto el CMS como Firebase poseerán una base de datos local, donde almacenarán la información correspondiente a las personas. Su fin es mantener la independencia entre los componentes, y de esta manera se obtiene una mejora en la performance, ya que la obtención de información de llevará a cabo localmente.

La responsabilidad de mantener ambas bases de datos consistentes es del CMS, ya que es el único medio por el cual se va a poder realizar modificaciones sobre la información de las personas perdidas.

Si bien, Firebase tiene a posibilidad de alterar dicha información, esta funcionalidad no es utilizada en el sistema. Solamente utilizará la información para poder ser brindada a los usuarios de las aplicaciones mobile.

Las direcciones de las flechas representan el flujo de la información y las comunicaciones entre los componentes. Por lo tanto, el componente CMS deberá hacer uso del componente del sistema de notificaciones para comunicarse con Firebase, sin poder realizar una comunicación directa entre ellos. En otras palabras, deben respetar las restricciones de comunicación impuestas por la arquitectura.

Para el caso particular de las aplicaciones mobile y el servidor de Firebase, al momento que los usuarios definen su ciudad de residencia, quedan subscriptos al tópico que esta representa. Esto representa un tipo de patrón arquitectónico llamado “Publish-Subscribe” donde lo usuarios se subscriben solo a los tópicos que son de su interés recibir este información, y cuando nueva información se encuentre disponibles para esos tópicos, los usuarios serán notificados inmediatamente.

La ventaja que conlleva adoptar este tipo de arquitectura, yace en que los usuarios solo accederán solamente a la información que les es útil en el momento en que se encuentre disponible para el tópico subscripto. Por lo tanto, cuando nuevas actualizaciones sucedan, el servidor será consiente de estos cambios y enviará las notificaciones correspondientes. Vale aclarar que para nuestro proyecto, los usuarios solo estarán subscriptos a un único tópico, ya que un usuario puede encontrarse físicamente en un único lugar.

1.2 Componentes del Sistema

Como se muestra en la arquitectura de la sección anterior, para el desarrollo del sistema se llevaron a cabo divisiones de responsabilidades en distintos componentes, con la finalidad de obtener un mayor grado de modularidad e independencia entre ellos. De dicho proceso de división, resultaron los siguientes componentes:

CMS (Component Management System): Se basará en una sección de administración donde su principal tarea es el ABM de aquellas personas que son denunciadas en los distintos organismos nacionales, como pueden ser las comisarias. Es fundamental una estricta organización, por medio de cuentas de usuarios privilegiadas, a aquellas personas autorizadas al acceso del sistema de administración. Dicha sección de administración radicará básicamente en un sitio WEB, donde se ingresará la información requerida de la persona ausentada para ingresarla al sistema o el reporte de aparición de alguna persona que se encontraba registrada. Como funcionalidad adicional, toda persona ajena al sistema tendrá la facultad de poder visualidad tanto las personas que se encuentran actualmente desaparecidas, como las que hayan sido encontradas.

Una vez que una nueva persona desaparecida es ingresada al sistema, el CMS tiene la responsabilidad de notificar al Web Service acerca de dicho alta de la persona al sistema, para así poder ejecutar el algoritmo de notificaciones y notificar a los usuarios.

Al momento de realizar los ABM de las personas desaparecidas, los datos proporcionados serán almacenados en una base de datos relacional local propia del CMS.

Sistema de Notificaciones: Este componte se encarga del procesamiento de información y la generación de notificaciones inteligentes hacia los usuarios del sistema. Las notificaciones van a seguir un modelo de expansión acorde a la distancia del lugar en donde se ausentó una persona, con el lugar en donde se encuentra actualmente un usuario del sistema. Se establecerá una velocidad de propagación de la notificación, de manera que aquellos usuarios que se encuentren a una gran distancia del lugar donde

se ausentó dicha persona no sean notificados inmediatamente, sino dentro de un periodo de tiempo tal que se evite excesivas notificaciones innecesarias.

Cuando desde el CMS se notifique que se realizó el alta de una nueva persona desaparecida, el sistema procesará dicha información y generará las notificaciones que correspondan. La tarea de generar las notificaciones consta en abrir una conexión con el servidor de Firebase e indicar a que tópico se debe realizar la notificación, adjuntando los datos necesarios de la persona desaparecida, para poder ser visualizada por los usuarios del sistema.

Servidor de Firebase: Su principal responsabilidad es la de mantener un registro de los tópicos y los usuarios inscriptos a ellos. En el momento en que el sistema de notificaciones envía los pedidos, Firebase de encarga de generar la notificaciones a dichos usuarios subscriptos a los tópicos.

Es el encargado de proporcionar la información de las personas desaparecidas a la aplicación mobile. Es por ello, que mantiene una base de datos NoSQL o no relacional propia, que en todo momento se mantiene consistente con la base de datos del CMS.

Aplicación Mobile: Su responsabilidad radica en recibir las notificaciones e informarle al usuario acerca de una nueva persona ausentada, como así toda la información vinculada con dicha persona extraviada. La aplicación también contará con una sección de personas desaparecidas y con mecanismos que generen alertas en caso de que algún usuario vea a la persona que se está buscando en ese momento, como así formularios y números de organismos para brindar información.

La plataforma sobre la que la aplicación Mobile se ejecutará es Android. Esto es debido a la cantidad de usuarios que hacer uso de esta plataforma. En el capítulo de Firebase, se indagará más acerca del tema haciendo uso de diferentes estadísticas y estudios.

1.3 Conclusión

Definir y diseñar una correcta arquitectura desde el principio del proyecto, evita que no ocurran problemáticas ni cambios a futuros sobre el código a implementar. De esta manera, se tiene un amplio panorama cerca de la forma en que se debe estructurar el sistema, preestableciendo una división de responsabilidades y alcanzando mayor grado de modularidad y abstracción.

Es deseable, que esta arquitectura se respete a lo largo de la implementación del proyecto, para así poder hacer uso de los beneficios que aporta.

Capítulo 2 - Frameworks para el Desarrollo de la Aplicación Web

Se introducirá aquí los conceptos básicos acerca de los frameworks para el desarrollo web como así el patrón arquitectónico MVC que muchos de estos utilizan. Se explicarán y clasificarán las herramientas candidatas para llevar a cabo la implementación del Component Management System (de ahora en adelante CMS), así como cuales fueron los criterios de evaluación para determinar la que mejor se ajuste para su desarrollo.

2.1 Framework

Los frameworks para el desarrollo de software surgen de la necesidad de los programadores de estructurar y proporcionar herramientas a los mismos, de manera qué facilite y agilice la creación de nuevas piezas de software.

Si bien no existe una definición formal para explicar que es un framework en el ámbito de la computación, podemos utilizar como referencia ciertas aproximaciones a ella. En el año 1994, Nelson Carl integrante de la organización IEEE Computer, desarrollo una aproximación a la definición del significado de un framework:

“A framework helps developers provide solutions for problem domains and better maintain those solutions. It provides a well-designed and thought out infrastructure so that when new pieces are created, they can be substituted with minimal impact on the other pieces in the framework.”

En castellano podría traducirse de la siguiente manera:

“Un Framework ayuda a los desarrolladores a proporcionar soluciones para el dominio de un problema, como así también a un mejor mantenimiento de estas soluciones. Brindan una estructura bien diseñada y pensada estructura, de modo que cuando se crean nuevas piezas de software, estas puedan ser sustituidas con un mínimo impacto en otras piezas del framework”

Entonces, clarificando lo dicho anteriormente, a través de un framework podemos obtener nuevas piezas de software solamente modificando o agregando nuevas piezas al mismo, y de esta forma tener una base para poder empezar un nuevo proyecto, reutilizando librerías y piezas de código ya implementadas.

2.2 Frameworks Webs

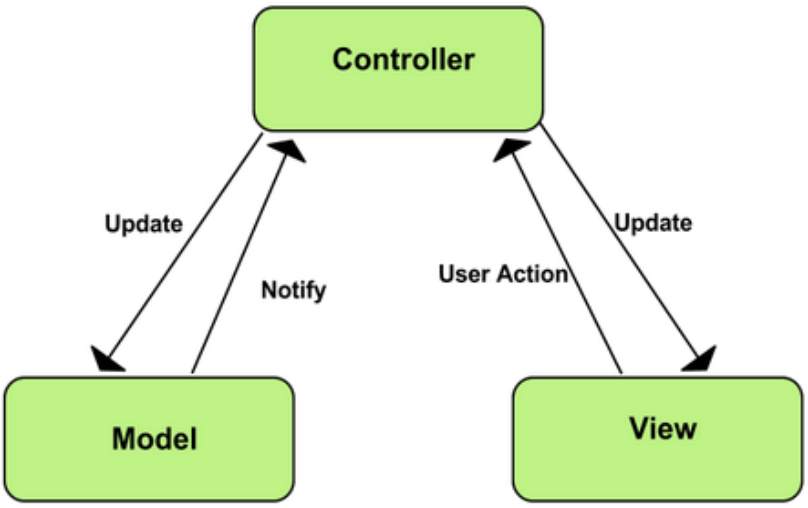
Una vez clarificado el concepto general de framework, podemos profundizar el mismo para definir frameworks Web. Para tal fin describiremos las características y estructuras necesarias para este tipo particular de framework.

Un framework Web, como su nombre lo indica, básicamente es un framework orientado al desarrollo de aplicaciones Web. Este tipo de frameworks pueden clasificarse de varias maneras, aunque de acuerdo a la bibliografía no hay un acuerdo sobre qué criterio utilizar. Los dos más comunes son el lenguaje de programación que utiliza y el otro el tipo de arquitectura subyacente.

Como particularidad, estos pueden estar implementados en distintos lenguajes de programación como lo pueden ser PHP, JavaScript, Python; y estos a su vez mantienen en su estructura, patrones particulares como lo puede ser Modelo-Vista-Controlador (ahora en adelante MVC). Cabe aclarar que es de suma importancia el lenguaje de programación que utiliza el framework Web, ya que en muchas casos, resulta decisivo

para la elección del mismo. Se detallará más información sobre dichos lenguajes en las próximas secciones.

El patrón arquitectónico MVC, es un patrón de arquitectura de software que realiza una separación entre los datos, la lógica de una aplicación de la interfaz de usuario, y el módulo encargado de gestionar los eventos y las comunicaciones. Como su nombre lo menciona, los tres componentes que definen al patrón son el modelo, el controlador y la vista. La interacción entre estos componentes se encuentra limitada, de manera que no interactúan todos entre sí, sino que el controlador es el encargado de establecer una comunicación entre el modelo y la vista.



Para comprender mejor el funcionamiento del patrón, se procederá a describir el funcionamiento individual de cada uno de los componentes:

Modelo: Su principal función es almacenar la representación de la información con la cual el sistema opera. Por lo tanto, gestiona todos los accesos a dicha información (consultas y actualizaciones), implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación. El modelo no posee información acerca de la vista y del controlador. Por lo general cuando el modelo cambia, esto es informado al controlador.

Controlador: Responde a eventos e invoca peticiones al modelo cuando se hace alguna solicitud sobre los datos almacenados. También puede enviar información a su vista asociada, si se solicita un cambio en la forma en que se presenta el modelo. Por lo tanto, se podría decir que el controlador hace de intermediario entre la vista y el modelo.

Vista: Representa la información almacenada en el modelo, en un formato adecuado para poder interactuar con el usuario. Esta información es procesada y proporcionada por el controlador. Cualquier cambio que se genere en base a acciones del usuario sobre la vista (como lo pueden ser el procesamiento de formularios, o ABM de entidades en la bases de datos), serán comunicados al controlador y este se encargará de informar los correspondientes cambios al modelo.

En resumen, el patrón MVC brinda modularidad a los desarrolladores de aplicaciones Webs y brinda grandes ventajas como la generación de código reutilizable y extensible, la separación entre la lógica de vista y la lógica de negocio, el trabajo simultáneo entre los desarrolladores que se encargan de diferentes componentes (como capa de interfaz de usuario y la lógica de la base), y la fácil mantención del mismo impulsada por la separación de dichos componentes.

2.3 Ventajas y Desventajas de un Framework Web

Al momento de realizar algún desarrollo web, resulta importante tener en cuenta las ventajas y desventajas que reportarían para el desarrollador el uso de algún framework. Es de suma importancia evaluarlas ya que la utilización de un framework no siempre será la mejor alternativa para el desarrollo de la solución buscada. El tamaño del proyecto y su grado de complejidad, el conocimiento de los desarrolladores acerca del framework a utilizar, y la facilidad para desenvolverse con ellos, pueden ser altamente determinantes para la elección.

Ventajas

El hecho de utilizar un framework para desarrollar una pieza de software, puede proporcionar numerosas ventajas.

Una de las más destacadas es la facilidad que le proporciona al programador para desarrollar código de manera rápida y sencilla, por medio de sus herramientas ya implementadas. Estas ayudan a resolver aquellas cuestiones que son básicas en las actividades cotidianas de un programador, como por ejemplo las conexiones con las bases de datos y el enrutado de los enlaces webs, entre otros. Como consecuencia, aumenta considerablemente la calidad del software que se obtiene, como así la mantención del código a futuro, ya que tiene una estructura modular para todos los proyectos, respetando las buenas prácticas de programación.

Al hacer uso del modelo MVC, explicado en la sección anterior, se estructura y facilita al programador un entorno de desarrollo, el cuál modula y encapsula por separado las responsabilidades de los distintos componentes.

Como se utiliza un framework para llevar a cabo el desarrollo, se proporciona una estructura estándar de forma tal que cualquier programador que tenga experiencia en el uso del mismo, pueda fácilmente entender y extender funcionalidad de una pieza de software ya creada.

Es de suma importancia destacar positivamente que tanto las funcionalidades, como todas las características de estos frameworks, se encuentran bien documentadas y explicadas para que puedan ser utilizadas por los distintos desarrolladores. Existen grandes comunidades que continuamente ayudan a dar soporte del mismo, y a su vez, implementan nuevas librerías y componentes para extender sus funcionalidades.

Por todo lo detallado anteriormente, es que un framework web es de suma importancia para el desarrollo de nuevas aplicaciones web. Es recomendable que aquellos proyectos que sean de menor escala y puedan ser fácilmente desarrollados sin esta herramienta, eviten el uso de la misma, ya que pueden resultar perjudicados al innecesariamente hacer uso obligado de ciertas funcionalidades.

Desventajas

Al igual que existen las ventajas de utilizar un framework web, estos cuentan con diversas desventajas que en ciertos casos resulta determinante en su elección. La falta de experiencia en el uso de framework por parte de los desarrolladores, la carencia de funcionalidades, la baja en el soporte para los mismos, son algunas de las cuestiones que impactan negativamente en ellos.

Entonces, tomando en cuenta lo mencionado anteriormente, podemos destacar que una de las más grandes desventajas radica en aquellos usuarios que no poseen experiencia en el uso de la herramienta, pudiendo resultarles difícil de entender y aprender las funcionalidades y características que estos brinda. Por ende, hasta que no se adquiera un conocimiento previo de su funcionamiento y estructura, no se llegará al desarrollo ágil de una pieza de software. Cabe destacar, que el proceso de aprendizaje es independientemente del conocimiento del desarrollador en el lenguaje de programación, dado que la complejidad se centra en la estructura general del proyecto y el funcionamiento de cada uno de sus componentes.

También existen frameworks que se encuentran en desarrollo, o frameworks cuyo potencial no es lo suficientemente amplio para abarcar los requerimientos del proyecto. Muchas veces esta problemática puede no ser detectada en el análisis previo al momento de la elección del mismo, resultando en un fracaso rotundo o en la necesidad de re implementar el proyecto con algún otro framework.

Como una de las mayores desventajas, existe la posibilidad de que el framework deje de proporcionar soporte y nuevas herramientas a los desarrolladores. En consecuencia, si se desea expandir el proyecto a futuro, puede que ciertos requerimientos del mismo no puedan ser implementados.

Teniendo en cuenta las desventajas analizadas se espera que, en una primera etapa, se evalúe correctamente si es necesario utilizar un framework web para el desarrollo de la aplicación, la popularidad y las contantes actualizaciones que estos brindan a los desarrolladores, como así los numerosos requerimientos que el proyecto posee. En caso de que esta evaluación sea no sea lo suficientemente completa, por lo explicado anteriormente, puede resultar perjudicial en el desarrollo, o en el peor caso, puede resultar en un fracaso.

2.4 Clasificación de los Frameworks Webs:

Hoy en día el desarrollo de nuevos frameworks webs, es una actividad muy frecuente dentro del área del desarrollo de software. Esta actividad se ha incrementado en los últimos años, permitiendo una contaste salida de nuevos frameworks al mercado. Estos pueden cumplir con alguno de los formatos de licencias libres o pueden ser de carácter privativo.

Están compuestos por numerosas características que particularizan a un framework a la hora de su elección. Si bien no existe una única forma de clasificarlos, en esta sección nos concentraremos únicamente en un subconjunto acotado de ellas, dado por contexto el proyecto que se desea desarrollar.

Por último, se clasificarán y desarrollaran brevemente los frameworks más importantes a la actualidad, teniendo en cuenta los grupos resultantes de la evaluación de las características de los mismos.

Análisis para la clasificación de los Frameworks Webs:

Como se desarrolló en la introducción de la sección, existen numerosas características sobre los frameworks webs. Entre estas características podemos mencionar el tipo de lenguaje que utilizan para el desarrollo, el patrón arquitectónico que toma como estructura principal, una posible división en módulos u orientación a objetos, la licencia con la que se encuentran patentados, entre otros.

Por lo tanto, analizando y teniendo en cuenta el contexto del proyecto a desarrollar, es que se realizará la clasificación basándose únicamente en el tipo del lenguaje y el patrón arquitectónico que el framework utiliza.

En lo que concierne al patrón arquitectónico, solo incluiremos un subconjunto acotado de frameworks que utilicen para su estructura interna el modelo MVC. Esta restricción se basada en que el patrón MVC es el que mejor se ajusta a las necesidades del proyecto, dada su simpleza como así la división y comunicación entre sus módulos. Su funcionamiento como su estructura, se encuentran explicados en la sección 1.2.

Por consecuente para este proyecto en particular, el factor que clasifica a los frameworks webs en los distintos grupos, es el lenguaje que utilizan. En el desarrollo web, existen numerosos lenguajes que estos adoptan para su implementación. Entre ellos podemos nombrar PHP, JavaScript, Phyton, Ruby, Java, HTML, entre otros. Tomaremos únicamente como referencia, los lenguajes PHP, JavaScript y Phyton, ya que son los lenguajes más utilizados en la actualidad.

Finalmente, obtenemos 3 grupos distintos sobre los cuales se clasificarán los frameworks webs. Estos serán denominados “Frameworks PHP”, “Frameworks JavaScript” y “Frameworks Phytom”.

Grupos para la clasificación de Frameworks Webs:

A continuación, se llevará a cabo una explicación detallada de cada uno de los grupos mencionados anteriormente:

Framework PHP: Este tipo de framework utiliza el lenguaje PHP para el desarrollo de nuevas piezas de software. Por lo general, son utilizados en casos donde la sobrecarga es mayor del lado del servidor y no tanto del lado del cliente. Por ende, se responsabiliza al servidor de generar las vistas y todo tipo de acciones y tareas sobre el modelo y el controlador. Son recomendados en aplicaciones donde la cantidad de las tareas es excesiva para el cliente o no es necesario trabajar de forma dinámica en el mismo.

Framework JavaScript: Estos se basan el lenguaje JavaScript como lenguaje principal. A diferencia de los frameworks PHP la sobrecarga es mayor del lado del cliente que por sobre el servidor. De esta forma, se dividen las responsabilidades y las cargas que generan las tareas haciendo un mayor aprovechamiento de los dispositivos de los clientes.

Son muy utilizados en caso de que se requiera mucho dinamismo en los sistemas web que se generan, evitando excesivas peticiones al servidor por cada acción que lleva a cabo los usuarios. Es decir, algunas cuestiones se resuelven rápidamente del lado del cliente sin necesidad de hacer un pedido al servidor. Entre algunas de estas acciones podemos destacar rápidos cambios de porciones de vistas, pre-procesamientos de formularios, requerimientos AJAX, etc.

Framework Python: Su lenguaje principal se basa en Python, donde fundamentalmente se trabaja con la programación orientada a objetos. Algunos frameworks Python poseen una arquitectura particular, como es el caso de Django, que utiliza un patrón arquitectónico denominado MVT (Modelo-Vista-Template), basado en MVC, pero que prescinde del motor de éste, es decir, del controlador.

Son frameworks muy robustos, pensados para el desarrollo de grandes aplicaciones. Por ello, debe considerarse la posibilidad de prescindir de ellos, cuando el proyecto a implementar sea una aplicación liviana o simple, ya que el consumo de recursos y el rendimiento no estarán compensado por la robustez del desarrollo.

Clasificación de los Frameworks Webs más populares.

Teniendo en cuenta los grupos descriptos en la sección anterior, es que se procederá a clasificar y desarrollar brevemente, los frameworks webs más populares en la actualidad.

Frameworks PHP: Dentro de esta categoría podemos encontrar 3 de los frameworks más utilizados en los últimos años. Ellos son “Synfony”, “Cake PHP” y “Laravel”.

Symfony:

Es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web por medio del patrón MVC. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony está desarrollado completamente con PHP y es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Posee numerosas características de las cuales podemos destacar:

- Su facilidad de instalar y configurar en la mayoría de plataformas, ya que posee un sencillo mecanismo de instalación como lo puede ser “Composer”.
- Independiente del sistema gestor de bases de datos, pudiendo elegir entre MySQL, PostgreSQL, Oracle y SQL Server de Microsoft.
- Brinda facilidades para poder ser utilizado tanto en pequeños como en grandes proyectos, dada la flexibilidad que otorga sus componentes.
- Adopta la mayoría de las mejores prácticas y los mejores patrones de diseño para la web, como lo es el patrón MVC.
- Se encuentra diseñado para aplicaciones empresariales, con la posibilidad de poder ser adaptable a las políticas y arquitecturas propias de cada empresa. Además, se caracteriza por ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Es fácil de extender, lo que permite su integración con librerías desarrolladas por terceros de una manera muy sencilla.

CakePHP:

Es un famework libre, de código abierto, para el desarrollo rápido de aplicaciones. Esta construido bajo el lenguaje PHP, brindando una estructura fundamental para ayudar a los programadores a crear aplicaciones web. El objetivo principal es permitir trabajar de forma estructurada y rápida, sin pérdida de flexibilidad. Pone a disposición todas las herramientas que se necesitan para poder desarrollar la lógica específica de la aplicación, haciendo uso del patrón MVC como estructura principal.

Se procederá a listar algunas de las numerosas características que posee dicho framework:

- Se encuentra patentado bajo licencias libres, siendo el caso de la “licencia MIT”.
- Brinda una amplia compatibilidad, tanto con las versiones más actuales, como para las versiones más antiguas de PHP.
- Hace uso de la herramienta “CRUD” para realizar la interacción de la base de datos, otorgando una mayor facilidad para la comunicación entre ambos componentes.

- Mecanismos de generación automática de código, tanto para el modelo como para vista y el controlador. Esto genera una mayor rapidez a la hora de desarrollar nuevas piezas de software.
- Utilización de patrón MVC como estructura principal del framework.
- Soporte para otras tecnologías como lo pueden ser AJAX, JavaScript, HTML, entre otros.

Laravel:

Es un framework de código abierto para desarrollar aplicaciones y servicios web. Se encuentra basado en el lenguaje PHP, utilizando en las versiones más recientes PHP 5. Su objetivo es desarrollar aplicaciones de forma elegante y simple. También ofrece las funcionalidades necesarias para que las aplicaciones desarrolladas sean lo más modernas y seguras. Está equipado con numerosas nuevas tecnologías, en las que se incluye enrutamiento RESTful, PHP nativo, eloquent ORM, entre otros.

Para su construcción se utilizaron varios componentes del framework Synfony. Por lo tanto, esto conlleva a que las aplicaciones web estén basadas en un código base confiable y optimizado, dado por los años de desarrollo que Synfony posee.

Entre algunas de las características de Laravel se incluyen:

- Posee un poderoso sistema de ruteo propio, con la posibilidad de poder ser alternado con el enrutamiento RESTful.
- Utiliza Blade como motor de desarrollo de plantillas, brindando mayor facilidad para construir las vistas.
- Para la interacción con la base de datos se utiliza Eloquent ORM, otorgando a los desarrolladores la capacidad de evitar escribir extensas consultas SQL y optimizando la construcción de las mismas.
- Una fácil y rápida Instalación haciendo uso de la herramienta “Composer”.
- La estructura interna del framework se encuentra basada en la arquitectura MVC.
- Se encuentra en constante desarrollo, por lo que libera permanentemente nuevas actualizaciones y nuevas funcionalidad que aumenta considerablemente la potencia del mismo.

Frameworks JavaScript: Para esta categoría, el framework más popular y utilizado es "AngularJS".

Angular JS:

Es un framework MVC de código abierto desarrollado por Google y escrito en Javascript, que trabaja del lado del cliente (client-side), permitiendo realizar de forma más dinámica nuestra aplicación web. Hace uso de otras tecnologías como HTML y CSS, así como librerías de terceros.

Este framework permite estructurar, organizar y escribir código de una manera más eficiente y en un tiempo menor, haciendo palicaciones más rápidas de acuerdo a la manera en la que evolucionan los motores de render de los navegadores. Esto significa que, entre más poderoso y eficiente un navegador web, aplicaciones web más rápidas e intuitivas se necesitan. Trabaja con el patrón MVC (Modelo, Vista, Controlador), lo que permite separar correctamente la lógica, el modelo de datos y la vista en una aplicación web.

AngularJS permite realizar aplicaciones de tipo SPA (Single Page Applications), lo que significa que podemos construir una aplicación web en donde una parte de la misma cargue dinámicamente, sin que se tenga que recargar todo la página. Esto permite hacer una aplicación web más rápida y fácil.

Algunos objetivos de diseño son:

- Disociar el lado del cliente de una aplicación del lado del servidor. Esto permite que el trabajo de desarrollo avance en paralelo, y permite la reutilización de ambos lados.
- Disociar la manipulación del DOM de la lógica de la aplicación. Esto mejora la capacidad de prueba del código.
- Considerar a las pruebas de la aplicación como iguales en importancia a la escritura de la aplicación. La dificultad de las pruebas se ve reducida drásticamente por la forma en que el código está estructurado.

Como consecuencia de estos objetivos, se guía a los desarrolladores a través de todo el proceso del desarrollo de una aplicación, es decir, desde el diseño de la interfaz de usuario a través de la escritura de la lógica del negocio, hasta los tests de pruebas.

Entre sus mayores ventajas, importar AngularJS a un proyecto existente es tan simple como hacer uso de una librería externa en formato ".js", ya que se encuentra preparado para adatarse a cualquier ambiente de trabajo.

Frameworks Phyton: Se presentará el framework Django, que posee mayor llegada a los desarrolladores en la actualidad. Estos tipos de frameworks se encuentran en absoluto crecimiento y son de lo más novedoso para el desarrollo Web.

Django:

Es un framework de desarrollo Web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como MVC. La meta fundamental de Django es facilitar la creación de sitios web complejos. Pone su énfasis en el reutilización, la conectividad y extensibilidad de componentes, para un rápido desarrollo. Python es el lenguaje principal utilizado para implementar las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

La distribución principal de Django también aglutina aplicaciones que proporcionan un sistema de comentarios, herramientas para sindicar contenido via RSS y/o Atom, "Páginas Planas" que permiten gestionar páginas de contenido sin necesidad de escribir controladores o vistas para esas páginas, y un sistema de redirección de URLs.

Algunas de las características de Django son:

- Un sistema de mapeador objeto-relacional, también conocido como ORM.
- Aplicaciones adaptables que pueden instalarse en cualquier página gestionada con Django.

- Un sistema incorporado de vistas genéricas que permite a los desarrolladores evitar reescribir la lógica de ciertas tareas comunes.
- Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
- Un despachador de URLs basado en expresiones regulares.
- Un sistema "middleware" para desarrollar características adicionales, como por ejemplo, componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF y soporte de sesiones.
- Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.
- Documentación incorporada accesible a través de la aplicación administrativa (incluyendo documentación generada automáticamente de los modelos y las bibliotecas de plantillas añadidas por las aplicaciones).

2.5 Elección del Framework para el desarrollo de la Aplicación Web

En las secciones anteriores, se trató sobre el concepto de framework, sus posibles clasificaciones y una pequeña descripción de los más utilizados en la actualidad. A continuación se procederá a destacar los aspectos más importantes del proyecto, el proceso de selección basándose en dichos aspectos, y las ventajas que hacen que “Laravel” sea el framework elegido para llevar a cabo el desarrollo.

Análisis de los aspectos más importantes para la Aplicación Web

Existen numerosos aspectos importantes a destacar, que son necesarios tener presente a la hora de realizar un análisis de la elección del framework. Entre ellos se pueden listar:

- El lenguaje de programación nativo del framework de desarrollo.
- La utilización del patrón arquitectónico MVC como estructura principal del framework.
- Determinar si la aplicación va a requerir mayor sobrecarga de tareas del lado del servidor o del lado del cliente, y evaluar la necesidad de páginas dinámicas.

- La facilidad de aprendizaje y la rapidez que brinda el framework a la hora de desarrollar nuevas aplicaciones.
- El grado de madurez del framework, junto con el soporte de las distintas comunidades que los avalan.
- La necesidad de incorporar conocimientos de nuevas herramientas, distintas de las aprendidas a lo largo de la carrera.

Proceso de elección del Framework

El proceso de selección del framework para el desarrollo del proyecto, estará basado haciendo uso de los puntos anteriormente mencionados. A continuación se realizará una breve explicación sobre aquellos frameworks que no se consideraron aptos:

Synfony es uno de los framework que cumple con todos los requisitos, y de esta forma, podría llegar a ser el que mejor se ajuste a las necesidades del proyecto. Entre estos requisitos podemos mencionar sus años de desarrollo y de mejoras, la gran comunidad de personas que ayudan a su evolución constante, el uso del patrón MVC, y su facultad de ejecutar las tareas del lado del servidor. La mayor desventaja radica en que durante el proceso de desarrollo de Laravel, se reutilizaron y mejoraron algunas herramientas de Synfony. Esto llevó a la construcción de un framework donde solo se implementaron

aquellos aspectos que se consideraron necesarios, ya sea por malas decisiones de diseño, baja calidad en el desarrollo, herramientas desactualizadas, entre otros; y así lograr una herramienta aún más sencilla, aumentando su facilidad y rapidez de comprensión y utilización.

El framework CakePHP es otro de los fuertes candidatos a la hora de la elección. Una de las razones por la que se descartó, es por el hecho de que dicho framework ya fue utilizado a lo largo de la carrera, teniendo un conocimiento previo. Por otro lado, la actualización de las últimas versiones no han sido del todo bien aceptadas por las grandes comunidades de desarrolladores, detectando grandes cantidades de fallos y

una gran disminuyendo la flexibilidad en relación con las versiones anteriores. Es por ello, entre algunas otras cuestiones, que el uso de CakePHP ha disminuido en los últimos años.

En este caso, son muchos los motivos por los cuales no se adopta a AngularJS para el desarrollo del proyecto. La principal característica por la cual se descarta, se basa en que se opta por una postura sobre el lado del servidor y no sobre el lado del cliente. Es decir, es necesario una estructura en el servidor sea el responsable de ejecutar las tareas y generar las vistas, e interactúe con los distintos Web Service que se van a necesitar para el funcionamiento del proyecto. Considerando que la aplicación tiene como objetivo ejecutarse sobre hardware que carece de últimas tecnologías, es de sumo interés que estos no tengan una gran sobrecarga. Por último, no es necesario contar con páginas que tengan un gran dinamismo, ya que se necesita una aplicación Web los más rápida y sencilla posible.

Con respecto al framework Django, este presenta excesivos consumos de recursos dada la simplicidad de la aplicación. Es decir, no se llega a compensar la robustez que otorga por sobre la sencillez que se necesita para desarrollar la aplicación Web. Es más recomendable su uso en aquellos casos donde el tamaño del proyecto sea lo considerablemente grande y complejo. Por otro lado, Django se diferencia de Laravel en el lenguaje en que se encuentran implementados. Se opta por un framework PHP dado a que en su esencia, PHP (Personal Home Page que aparece en reemplazo del lenguaje PERL), es uno de los lenguajes más elegido por los desarrolladores para el desarrollo web. Se destaca del mismo su gran compatibilidad con muchas plataformas, sus numerosas documentaciones y sus constantes nuevas versiones.

Ventajas del Framework Web Laravel

Dadas las comparativas realizadas por en la sección anterior, es que el framework elegido del proceso de selección, es Laravel. El mismo cuenta con numerosas ventajas y funcionalidades, que lo destacan de los demás framework y resulta ser el ideal para el dominio del problema que se desea implementar.

La potencia del framework radica en su amplia capacidad de reducción de costos y tiempos para el desarrollo y mantenimiento de piezas de software. Esta capacidad puede llegar a ser lograda gracias a las diversas herramientas integradas en el framework, junto con las facilidades y sencillez que estas brindan a la hora de ser utilizadas. Es por ellos, que se considera que Laravel posee una curva de aprendizaje relativamente baja.

Como se mencionó anteriormente, la estructura interna de Laravel se encuentra basada en el patrón arquitectónico MVC, adquiriendo todas las ventajas que el uso del mismo conlleva. Para aumentar la flexibilidad, adaptabilidad y reducción de código, es que el framework posee un sistema propio de “Routes With Clousures”. Básicamente el desarrollador posee la facultad de controlar todo el sistema de enrutado que se necesita para llevar a cabo la comunicación entre las vistas y los controladores.

Es de suma importancia tener presente la cantidad y calidad de la documentación del framework, ya que gracias a ella, es que los desarrolladores van a poder aprender a hacer uso del mismo con mayor rapidez. En este punto, Laravel posee una excelente y abundante documentación, no solo en su sitio oficial, sino también en los diversos foros y sitios de las distintas comunidades que lo conforman.

Su modularidad junto con su amplio sistema de paquetes y drivers, llevan a poder extender sus funcionalidades básicas de manera fácil, robusta y segura, por los desarrolladores del sistema. Cabe destacar que es una característica muy deseable de los frameworks, ya que a medida que surgen nuevos requerimientos en el proyecto, es que se encuentran en la necesidad de agregar nuevas funcionalidades.

Una de las herramientas que posee un alto nivel de importancia es el uso de Eloquent. Por medio de ella es que se aumenta el nivel de abstracción entre la base de datos que componen a los distintos modelos y los controladores del sistema. Esto hace que el manejo de los datos en Laravel no sea complejo y la interacción con las bases de datos es totalmente orientada a objetos. Como consecuencia, resulta compatible con la gran mayoría de las bases de datos del mercado actual y facilita la migración de datos. Por último, el uso de esta herramienta permite la creación de consultas robustas, complejas

y optimizadas, si necesidad de escribir código en alguno de los lenguajes de consultas de base de datos.

Posee su propio manejador de plantillas llamado “Blade”, que trae consigo la generación de mejoras en la sección de presentación de la aplicación, como la generación de plantillas más simples y organizadas. Además incluye un sistema de cache que mejora la rapidez de las mismas, y por consecuencia, el rendimiento de la aplicación.

Para mejorar la automatización a la hora del desarrollo, se cuenta con una herramienta de interfaces de líneas de comando llamada “Artisan”. Esta permite ejecutar tareas programadas como lo pueden ser las migraciones de base de datos, casos de testing, y compilaciones en general; como así la generación de nuevos modelos, vistas y controladores que el desarrollador requiera.

Para finalizar, vale destacar que existen otras ventajas dadas por las distintas funcionalidades y herramientas que componen al framework. En nuestro caso, no son de suma importancia mencionarlas, ya que, el dominio del sistema no requiere hacer uso de ellas.

2.6 Conclusión

Como se puede apreciar, en la actualidad existen una gran cantidad de frameworks para el desarrollo de aplicaciones Webs. Dependiendo del tipo de aplicación que se desea llevar a cabo en conjunto con sus características, es que se requiere un proceso de elección del framework a utilizar. Una mala elección del mismo, puede llevar a un rotundo fracaso y/o re implementación haciendo uso de algún otro framework.

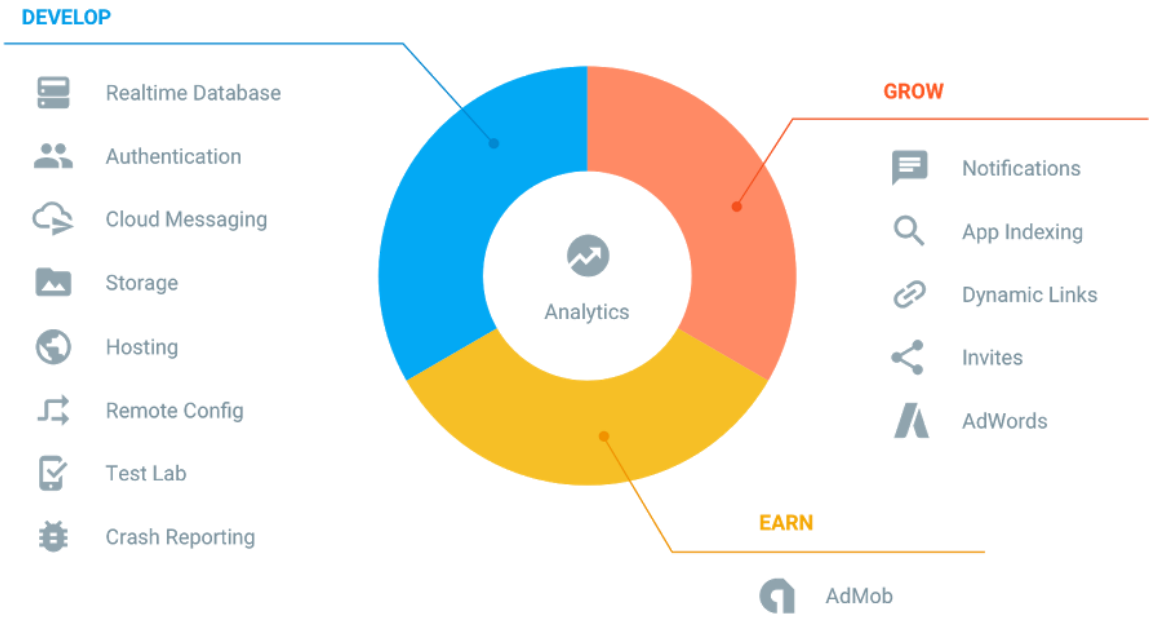
Es por ello que se requiere en todo momento, tener presente el tipo de arquitectura que se va a adoptar, la necesidad de una manipulación constante de la información, evaluar el grado de dinamismo que la aplicación requerirá, y el hardware que poseerá tanto los clientes como el servidor.

Como punto a favor, el hacer uso de dichos frameworks aporta a los desarrolladores una herramienta que facilita la rápida construcción de nuevas aplicaciones Webs, donde el producto final posee un alto grado de calidad y extensibilidad a futuro.

Capítulo 3 - Firebase

En el siguiente capítulo se presentará la herramienta Firebase, utilizada para el desarrollo de aplicaciones móvil.

Firebase es una plataforma móvil creada por Google, cuyo objetivo principal es el desarrollo de aplicaciones para móviles. Está especialmente dirigido a aplicaciones de negocios, con la intención de ayudar a las empresas a incrementar sus bases de usuarios y aumentar sus ganancias a través de dichas aplicaciones móviles. No obstante, sirve para el desarrollo de emprendimientos y proyectos de pequeña escala. La captura de pantalla que se presenta a continuación, muestra sólo algunas de las características que son parte de Firebase. Entre ellas podemos notar las intenciones de Google por brindar a nuestro proyecto facilidades para desarrollar nuevas características, un constante crecimiento y la posibilidad de generar ganancias.



Esta plataforma sirve para el desarrollo del Back-End de nuestra aplicación. En otras palabras, Firebase provee a los desarrolladores herramientas ya implementadas, para que en base a ellas, puedan realizar desarrollar las funcionalidades de sus proyectos. De esta forma, fácilmente se pode enfocar en el Front-End de la aplicación sin tener conocimientos del funcionamiento de sus estructura interna alojado en la “Nube” de Google.

La centralización de todas estas herramientas en una única consola, brinda mayor flexibilidad y facilidad a los desarrolladores, ya que exenta a los mismos de tener que utilizar diferentes herramientas, que en algunos casos, presentan incompatibilidades entre sí. Se destaca la sencillez en la incorporación que brinda, tanto para nuevos proyectos como para proyectos existentes, por medio de la importación de sus librerías online.

Por último, Firebase está pensado para el desarrollo de aplicaciones móviles en diversas plataformas, como Android y IOS. Existen adaptaciones de Firebase para el desarrollo de aplicaciones webs bajo algunos frameworks, como por ejemplo AngularJS.

En las siguientes secciones se comentaran las distintas componentes que la conforman, sus objetivos y finalidades. También se presentaran las ventajas y facilidades que su uso aportan a los desarrolladores.

3.1 Componentes de Firebase

Con Firebase, se puede enfocar el tiempo y atención en el desarrollo de la mejor aplicación posible. Dado que las operaciones y funciones internas son sólidas y están atendidas por la interfaz de Firebase, se puede pasar más tiempo en el desarrollo de la aplicación de alta calidad para que puedan ser utilizadas por los usuarios. Esto es posible gracias a los componentes que lo conforman, donde los más destacados son:

Firebase Cloud Messaging - Notifications (Mensajería en la Nube - Notificaciones): Firebase Cloud Messaging (de ahora en adelante FCM) es una solución multiplataforma que permite enviar de forma segura, mensajes y notificaciones. Al usar FCM, se permite notificar a una aplicación del cliente acerca de nuevos correos electrónicos u otros datos que estarán disponibles para la sincronización. Se puede enviar notificaciones para aumentar la receptación y retención de usuarios. Para casos de uso como mensajería instantánea, un mensaje puede transferir una carga de hasta 4 KB hacia la aplicación del cliente.

La implementación de FCM incluye un servidor que interactúa con la aplicación móvil, a través del protocolo de HTTP o XMPP. Además, FCM incluye una consola de notificaciones, que puedes usar para enviar notificaciones manualmente.

Firebase Notifications es un servicio de notificaciones de Google que está basado en FCM. Comparten el mismo “Software Development Kit” (conocido como SDK) para el desarrollo de aplicaciones móviles, y es usado en casos donde se requiere notificaciones simples. Esto es, el caso de necesitarse enviar mensajes de captación o marketing con perfilamiento integrado y/o análisis de datos. Para implementaciones con requisitos de mensajería más complejos, es requiere el uso de FCM.

Firebase Authentication (Autentificación de Usuarios): En la mayoría de las aplicaciones, por lo general, se requiere conocer la identidad de un usuario. El reconocimiento de la identidad de un usuario permite a una aplicación guardar datos de éste de manera segura en la nube, y brindar la misma experiencia personalizada en todos sus dispositivos. Proporciona servicios de Back-End, por medio de SDK fáciles de usar y bibliotecas de IU ya implementadas, para poder llevar a cabo el proceso de autentificación de usuarios. Admite autenticación por ingreso de contraseñas, como así otros proveedores de identidades populares entre los que podemos destacar Google, Facebook y Twitter.

Firebase Authentication se integra estrechamente con otros servicios de Firebase y aprovecha estándares industriales como OAuth 2.0 y OpenID Connect. Por lo tanto, es fácilmente integrable con Back-Ends previamente desarrollados.

Firebase Realtime Database (Base de Datos en Tiempo Real): Las aplicaciones almacenan y sincronizan datos en las bases de datos NoSQL alojada en la nube de Google. Los datos se sincronizan con todos los clientes en tiempo real y estos seguirán estando disponible incluso cuando la aplicación pierda la conexión a internet.

Estas bases de datos son consideradas NoSQL, ya que los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. Si la aplicación soportara multiplataforma (como puede ser iOS, Android y los SDK de JavaScript), todos los clientes compartirán una única instancia de las bases de datos en tiempo real, y recibirán actualizaciones de forma automática con los últimos datos disponibles.

Firebase Storage (Almacenamiento en la Nube): Fue creado para desarrolladores de aplicaciones móviles que necesitan almacenar y proporcionar contenido generado por el usuario, como fotos o videos. Firebase Storage ofrece la posibilidad de subir y descargar archivos de forma segura, independientemente de la calidad de la red. Puede ser usado para almacenar imágenes, audio, video y otro contenido generado por el usuario. Firebase Storage está respaldado por Google Cloud Storage, un servicio potente y simple de almacenamiento de objetos.

3.2 Firebase para Multiplataformas

Esta herramienta fue diseñada para el desarrollo de aplicaciones multiplataformas, entre las cuales podemos destacar Android, IOS, y Web. Su objetivo radica en poder hacer uso de la misma, independientemente de la plataforma que el usuario haga un uso frecuente.

3.3 Android como Plataforma de Desarrollo

Si bien Firebase brinda numerosa plataformas para desarrollar aplicaciones, las cuales fueron nombradas anteriormente, Android es la elegida para la implementación de la aplicación móvil. La principal razón que determina la elección, es la cantidad de usuarios a nivel país que poseen dispositivos que hacen uso de éste sistema operativo.

Según un informe emitido en el 2015 por “eMarketer” titulado “Android Extends Latin America Dominance to Tablets”, en Argentina el 75% de los usuarios poseen un dispositivos móviles con sistema operativo Android, contra tan solo el 11% de los usuarios que poseen un dispositivos móviles con sistema operativo IOS.

Entonces, dando en evidencia los datos anteriores y considerando que se quiere llegar a la máxima cantidad de usuarios finales, es que se eligió Android como plataforma única y principal de desarrollo.

3.4 Ventajas de hacer uso de Firebase

El potencial que posee la herramienta para la implementación de nuevas aplicaciones móviles, aportan un gran número de ventajas al proyecto a desarrollar.

Como una de las principales ventajas podemos destacar la incorporación de una base de datos en tiempo real, donde se almacena la información correspondiente a las personas que se encuentran actualmente desaparecidas. Se considera de tiempo real, ya que todos los cambios que se realicen sobre la misma, serán visibles en los dispositivos móviles instantáneamente. Por lo tanto, al hacer uso de ellas no requerimos ningún sistema de actualización constante, solo basta con indicar en la aplicación móvil que cuando hayas nuevos cambios, estos sean mostrados al usuario.

Su implementación se basa en la tecnología “NoSQL” que abarca una amplia gama de tecnologías y arquitecturas, buscando resolver los problemas de escalabilidad y

rendimiento de “Big Data” que las bases de datos relacionales no fueron diseñadas para abordar.

Otra ventaja de hacer uso de Firebase es su sistema de notificaciones. Este cuenta básicamente con 3 medios de notificaciones: por dispositivo individual, por tópico/tema, por segmento de usuario. Dependiendo de la modalidad elegida, se podrá notificar a un único usuario, a un grupo de usuario que estén subscriptos a un cierto tópico/tema, o a un segmento de usuarios que esté compuesto por conjunto específico de ellos.

Para esta aplicación en particular, los usuarios se subscribirán a un cierto tópico que representará su ciudad de residencial actual. Por ende, en caso que se genere una notificación a dicho tópico, todos los usuarios que estén subscriptos a él serán avisados inmediatamente.

Al hacer uso de Firebase Authentication, nuestro sistema automáticamente se desvincula del proceso de acceso de los usuarios, y como ventaja, se evita la implementación de un sistema propio de autentificación. Los datos de las cuentas de los usuarios serán manipulados por Firebase siendo totalmente transparente para nuestra aplicación. La misma solo deberá comunicarse con sus servidores y proporcionar la información que los usuarios brindar para el acceso. Existen 3 modos de autentificación, Google +, Facebook y Twitter. La modalidad que se eligió para la autentificación del proyecto fue Google +, dado que por poseer un dispositivo con sistema operativo Android, todos los usuarios poseerán una cuenta de E-mail de Google.

En conclusión, el hacer uso de Firebase brinda numerosas ventajas en lo que respecta a facilidades para el programador, como mejor aprovechamiento de tiempo y recursos, y ventajas a nivel de aplicación, ya que se puede enfocar mayormente en el desarrollo de sus funcionalidades y así lograr un producto más simple, robusto y eficiente.

3.5 Comunicación Laravel y Firebase

Dada la implementación y forma de trabajo de Firebase explicado en las secciones anteriores, se sabe qué hace uso estricto de una base de datos de tiempo real. Por otra parte, la aplicación Web en su implementación mantiene su propia base de datos local donde se registran las nuevas personas desaparecidas.

Es de suma necesidad mantener estas 2 bases de datos en todo momento para que ambos sistemas puedan llevar acabo sus funcionalidades. Es por ello, que la aplicación Web tiene la responsabilidad de comunicar a la base de datos de Firebase acerca de los cambios que sucedan en su base de datos local.

Para poder llevar acabo esta tarea, es que se utiliza una librería desarrollada exclusivamente para Laravel. Por medio de la misma, la aplicación Web puede realizar cambios en la base de datos de Firebase cada vez que existan cambios en su base de datos local, y de esta manera, poder mantener ambas bases de datos consistentes.

A modo de aclaración, la aplicación móvil no genera ningún cambio sobre la base de datos de Firebase, es decir, solamente la utiliza para recopilar la información acerca de las personas desaparecidas y mostrarlas a los usuarios en un formato adecuado. Es por ello, que no se requiere reportar cambios en las bases de datos desde Firebase hacia la aplicación Web.

Como ventaja de poseer la información duplicada en 2 bases de datos, en caso de que alguna presente inconsistencia en la información almacenada, podrá utilizarse la restante base de datos como “respaldo” de la información.

3.6 Conclusión

Para este proyecto en particular, hacer uso de Firebase aporta numerosos beneficios para poder alcanzar sus objetivos. Su sistema de notificaciones en conjunto con su base de datos en tiempo real y su sistema de autentificación, desvinculan a la aplicación móvil de funcionalidades que no son propias de la misma, sino que son generales a todos los proyectos.

De esta manera, se hace una reutilización de código importando estas librerías y se les permite a los desarrolladores enfocarse solo en aquellas cuestiones propias del proyecto.

Desde el inicio de Firebase hasta el momento, se han desarrollado numerosas actualizaciones y nuevas funcionalidades, las cuales en un futuro podrán permitir expandir la aplicación para poder atender las nuevas necesidades de los usuarios.

Capítulo 4 - Sistema de Notificaciones

En el siguiente capítulo, se desarrollará la explicación del algoritmo empleado para poder llevar acabo el sistema de notificaciones, en conjunto con la estructura de datos que el mismo utiliza y los tiempos de ejecución resultantes. Adicionalmente se presentarán alterativas de estructuras que se contrastarán y evaluarán para alcanzar a la elección de la misma.

4.1 Objetivo

Las notificaciones son la herramienta que posee el sistema para informar a los usuarios que existen nuevas personas desaparecidas. Cuando una nueva persona es ingresada en el sistema se disparará un alerta a todos los usuarios, de forma tal, que primero sean notificados aquellos usuarios que pertenezcan al mismo lugar que la persona que desapareció. Luego, esta alerta se expandirá a todas las ciudades limítrofes, y así sucesivamente hasta que abarque todas las ciudades del país. En caso de que la persona sea encontrada, la alerta es finalizada.

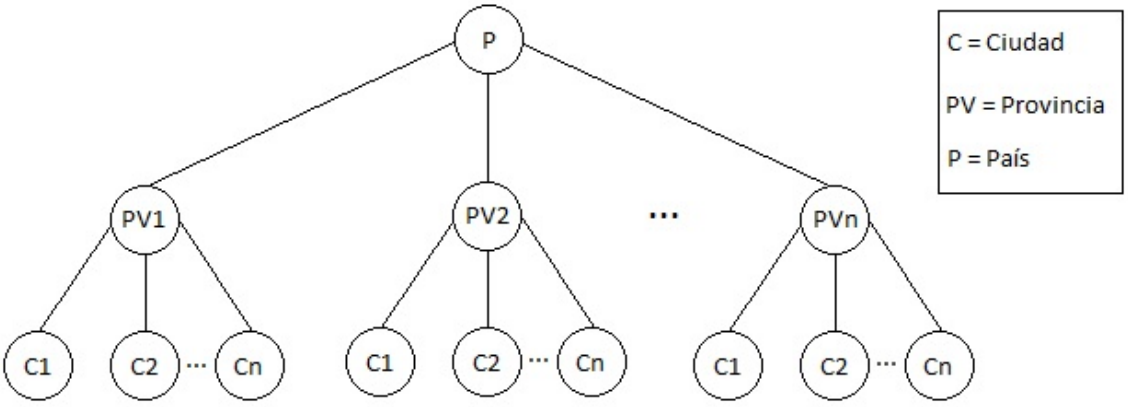
Es por ello que se requiere un algoritmo que simule ese comportamiento, y de esta manera, evitar que se notifique inmediatamente a usuarios que se encuentran a una gran distancia del lugar del hecho. De esta forma, se postergaría la notificación un tiempo proporcional al tiempo que se estima que la persona viaja del lugar del hecho, al lugar en donde reside el usuario.

4.2 Representación de la Estructura de Datos

Para poder llevar acabo la elección del algoritmo, es necesario tener en cuenta las distintas estructuras con las cuales se pueden representar los datos requeridos. Una correcta elección de dicha estructura, influirá directamente sobre la complejidad del algoritmo, su tiempo de ejecución y su correcto desempeño.

Estructura de Árbol

Como primera aproximación, se pensó en una estructura de datos en forma de “Árbol”, donde las ciudades se encuentran representadas como nodos hoja, y su raíz, como el país al que pertenecen las ciudades. Estas ciudades, a su vez se agrupan en distritos, los distritos en provincias, las provincias en regiones, y las regiones en un único país. Por ende, una posible representación resultaría:



Para ésta estructura, los límites entre las ciudades están representados por una relación de herencia entre un nodo padre y todos sus nodos hijos, es decir, una ciudad representada con el nodo “X” es limítrofe con otra ciudad representada con el nodo “Y” sí ambas son hijas del distrito representado con el nodo “Z”. Esta misma definición de límite se aplica tanto para los distritos, provincias y regiones.

Como es necesario tener conocimiento del contexto donde se va a ejecutar el algoritmo para poder llevar acabo la elección de la estructura, para este proyecto en particular, el contexto se encuentra conformado por ciudades, donde cada una de ellas se encuentra en cercanía con otras ciudades, las cuales son denominadas ciudades limítrofes.

Si bien, un país puede tener diferentes divisiones internas como las que representa la estructura de datos árbol, estas no son de relevancia para el algoritmo ya que solamente basta con que las notificaciones sean transmitidas de ciudad en ciudad limítrofe. Por lo

tanto, se necesita hacer uso de una estructura donde se encuentren representadas solamente las ciudades con sus respectivos nombres, junto con aquellas ciudades limítrofes a dicha ciudad.

Por lo tanto, esta forma de representación de las ciudades limítrofes conlleva a una gran problemática. Suponiendo que contamos con 2 ciudades A y B que son limítrofes entre sí, pero pertenecen a distintas provincias. Esto conlleva que, el distrito padre de la ciudad A será distinto al distrito padre de la ciudad B.

En el caso en que la notificación se genere desde la ciudad A, serán informadas todas aquellas ciudades limítrofes a dicha ciudad, es decir, que tengan el mismo distrito como distrito padre. Por lo dicho anteriormente, como las ciudades A y B no pertenecen al mismo distrito padre, la ciudad B no será notificada inmediatamente.

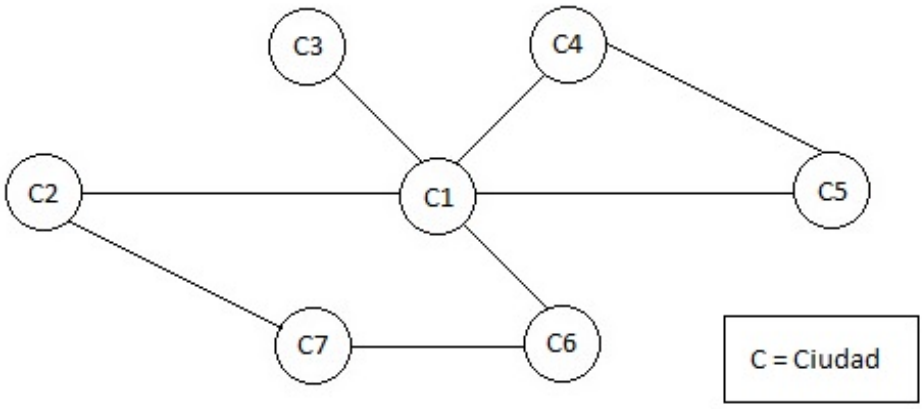
Luego, serán notificadas todas las ciudades de los distritos limítrofes al distrito de la ciudad A. Una vez que todos los distritos de la provincia sean notificados, se procederá a notificar las ciudades de las provincias limítrofes en la cual se encuentra incluida la ciudad B. En conclusión, el tiempo que tarda en notificarse a la ciudad B limítrofe a la ciudad A **es inapropiado**, ya que debería de haberse notificado inmediatamente y no una vez que se haya notificado a todas las ciudades de la provincia a la que pertenece la ciudad A.

Dicho comportamiento de la estructura no cumple con el objetivo del sistema de notificaciones, por lo tanto, no se considera apropiada para la implementación del algoritmo.

Estructura de Grafo

Viendo las problemáticas que se plantearon con respecto a las ciudades y sus ciudades límites, es que se pensó como alternativa una estructura de datos de “Grafo”.

En este tipo de estructura, las ciudades son representadas como “vértices” del grafo, y conectadas por “arcos”. El grafo mantiene en todo momento una lista con todos los vértices. Cada vértice a su vez posee una lista de los arcos utilizados para conectarse con los otros vértices. Por ende, se representa a las ciudades limítrofes a una ciudad por medio de los arcos que inciden de un vértice V a los demás vértices. Este tipo de estructura no representa ningún tipo de división. Una posible representación sería:



Como podemos apreciar, en caso de que se genere una notificación para una ciudad C representada por un vértice V, solo basta con notificar a todos los vértices adyacentes al vértice V. Luego, serán notificados todos los vértices adyacentes de aquellos vértices adyacentes al vértice V, y así sucesivamente hasta haber notificado a todos los vértices del grafo.

Por lo tanto, visto y considerando que se respeta el comportamiento de las notificaciones sobre la estructura con el planteado en los objetivos del sistema de notificaciones, es que se considera como la estructura elegida para llevar a cabo el algoritmo.

4.3 Implementación del Algoritmo de Notificaciones

En la sección anterior, se llevó a cabo el proceso de elección de la estructura de datos que va a ser utilizada para la implementación del algoritmo. La importancia de su correcta elección, influye directamente sobre la complejidad y el tiempo de ejecución del algoritmo.

Por lo tanto, el algoritmo básicamente se desarrollará como un recorrido sobre el grafo de forma tal que emita las notificaciones necesarias, cumpliendo con las restricciones que los objetivos imponen.

Introducción a la Estructura de datos

El objetivo principal del algoritmo es notificar a todos los usuarios del sistema que una persona se encuentra desaparecida, generando una notificación desde el lugar en donde se produce el hecho hacia todas las ciudades del país.

Como se especificó anteriormente, estas ciudades son representadas como vértices del grafo cuyos atributos son:

- El nombre de la ciudad.
- Una lista de las ciudades limítrofes.
- Un estado el cual determinará si la ciudad ya fue notificada o no.

Estos vértices son almacenados en una lista que se mantiene como atributo del grafo. Cada vez que se requiera realizar cualquier tipo de manipulación sobre ellos, es responsabilidad del grafo atender dichas peticiones, como por ejemplo obtener el nombre de una ciudad o sus ciudades limítrofes.

Recorrido BFS

Para llevar a cabo dicha tarea, es necesario realizar un recorrido sobre todos los vértices del grafo, respetando algún tipo de heurística. Haciendo provecho de la estructura y de la forma en que se requiere realizar el recorrido sobre los vértices, se determina que la mejor manera es realizar un recorrido BFS (Breadth First Search).

En un recorrido BFS los vértices son recorridos a lo ancho. Esto significa que, en una primera instancia se visita a todos los vértices adyacentes de un cierto vértice; luego, para cada uno de los nodos adyacentes se visitan sus respectivos nodos adyacentes, y así sucesivamente hasta alcanzar la totalidad de los vértices del grafo.

Por teoría de grafos, se conoce que el orden de tiempo de ejecución del algoritmo es considerado como O(|V| + |A|), donde |V| representa el número de vértices y |A| representa el número de arcos del grafo.

Explicación del Algoritmo de Notificaciones

El algoritmo del sistema de notificaciones será implementado en base al recorrido BFS, modificándolo según las conveniencias del problema.

Como primer paso del algoritmo, se obtiene el nombre de la ciudad en donde se produjo la desaparición de la persona. En base a dicho nombre de la ciudad, se pedirá al grafo el vértice inicial VI sobre el cual se comenzará el recorrido.

Una vez obtenido el vértice VI, se ejecutará como caso inicial:

1. Enviar por medio del sistema de notificaciones de Firebase (explicado en el capítulo X) una notificación a todos los usuarios inscriptos al tópico que coincida con el nombre del atributo del vértice VI.
2. Se almacenarán en una estructura auxiliar todos los vértices adyacentes del vértice VI.
3. Se establecerá el estado del vértice VI como “visitado”.
4. Esperar un determinado tiempo.

Luego, mientras existan vértices en la estructura auxiliar, se procederá a:

1. Se creará una segunda estructura auxiliar vacía.
2. Para cada vértice de la primer estructura auxiliar
   Si cuyo estado sea “por visitar”
   - Notificar a todos los usuarios inscriptos al tópico que coincida con el nombre del atributo del vértice V.
   - Pedir al grafo todos los adyacentes del vértice V.
   - Almacenar los vértices adyacentes obtenidos en la segunda estructura auxiliar.
   - Cambiar el estado del vértice V como “visitado”.
3. Reemplazar todos los vértices de la primera estructura auxiliar, por el total de los vértices de la segunda estructura auxiliar.
4. Esperar un determinado tiempo.

Finalmente, una vez que no queden vértices en la estructura auxiliar el algoritmo ha finalizado de emitir las notificaciones a todos los usuarios cuyos tópicos coincidan con los nombres de los vértices del grafo. En otras palabras, todos los usuarios de las distintas ciudades han sido notificados acerca de la desaparición de una persona en una cierta ciudad, y de manera, se respetaron todos los objetivos planteados para el sistema de notificaciones.

Pseudocódigo del Algoritmo de Notificaciones

A continuación, se presentará un pseudocódigo del funcionamiento del algoritmo descripto en la sección anterior:

```
Function doNotification(String nombreCiudad)
{
	Vertice VI = Grafo.getVertice(nombreCiudad);
	sendNotificationToTopic(VI.getNombre());
	arrayVisitar.put(Grafo.getVerticeAdyacentes(VI));
	Grafo.setVerticeVisitado(VI);
	Sleep(time);

	While (!Empty(arrayVisitar))
	{
		arrayVisitarNuevos = new array();
		ForEach (Vertice V in arrayVisitar)
		{
			If (!V.estaVisitado())
			{
				sendNotificationToTopic(V.getNombre());
				arrayVisitarNuevos.put(Grafo.getVerticesAdyacentes(VI));
				Grafo.setVerticeVisitado(VI);
			}
		}
		arrayVisitar = arrayVisitarNuevos;
		Sleep(time);
	}
}
```

4.4 Tiempo de ejecución del Algoritmo de Notificaciones

Como se mencionó en la sección 3.3, el algoritmo de notificaciones respeta la implementación básica del recorrido BFS con las modificaciones necesarias para poder llevarse a cabo de acuerdo al contexto del proyecto.

Tomando en cuenta que se está tratando con un grafo bidireccional o no dirigido, se sabe que van a existir 2 arcos por cada par de vértice del grafo y que la cantidad de vértices del mismo es equivalente al número de ciudades de un cierto país.

Como el grafo depende directamente del contexto, y este varía según el país en el cual se representar, es que pueden existir 3 variantes diferentes:

1) La cantidad de ciudades representadas por los vértices del grafo pueda ser mucho mayor que la cantidad de arcos que conectan a dichas ciudades, por ende, se concluye que |V| >> |A|.

2) La cantidad de ciudades representadas por los vértices del grafo pueda ser mucho menor que la cantidad de arcos que conectan a dichas ciudades, por ende, se concluye que |V| << |A|.

3) La cantidad de ciudades representadas por los vértices del grafo pueda ser igual o muy aproximada que la cantidad de arcos que conectan a dichas ciudades, por ende, se concluye que |V| = |A|.

Por lo tanto, podemos hacer uso de la del tiempo de ejecución del algoritmo BFS, para determinar el tiempo de ejecución nuestro algoritmo de notificaciones. Como se planteó en la sección 3.2, dicho tiempo de ejecución es equivalente a O(Max (|V|,|E|)).

Entonces, dependiendo del grafo derivado directamente del contexto en conjunto con su variante, es que el tiempo de ejecución del algoritmo podrá resultar en el orden de O(|V|) o en el orden de O(|E|).

4.5 Conclusión

Se desarrolló un algoritmo acorde a los objetivos que se especificaron para el sistema de notificaciones del proyecto. La elección de la correcta estructura de datos que mejor se amoldó a la representación del contexto, resultó en un beneficio directo a la hora de la implementación del algoritmo, ya que, se pudo hacer uso de algoritmos existentes como el BFS.

El orden del tiempo de ejecución resultante para el algoritmo se considera aceptable, lo que conlleva a que el servidor de la aplicación no se encuentre sobrecargado a la hora de ejecutar dicho algoritmo.

Capítulo 5 - Simple Object Access Protocol (SOAP) y WSDL

Se introducirá en este capítulo la noción del protocolo de comunicación SOAP, en conjunto con su funcionamiento y estructura. Se planteará las razones por las cuales fue elegido y cuál es su finalidad de uso dentro del proyecto. También, se desarrollará brevemente la tecnología WSDL utilizada para la manipulación de los XML's.

SOAP

Actualmente un sin fin de empresas se han decantado por el desarrollo de aplicaciones que puedan trabajar sobre Internet porque permite la distribución global de la información. Las tecnologías más usadas para el desarrollo de estas aplicaciones, han sido hasta hace poco CORBA, COM y EJB.

Cada una proporciona un marco de trabajo basado en la activación de objetos remotos mediante la solicitud de ejecución de servicios de aplicación a un servidor de aplicaciones. Estas tecnologías han demostrado ser muy efectivas para el establecimiento de sitios Web, sin embargo presentan una serie de desventajas, como son total incompatibilidad e interoperabilidad entre ellas, total dependencia de la máquina servidora sobre la que corren, así como el lenguaje de programación.

Esto ha llevado a la necesidad de considerar un nuevo modelo de computación distribuida de objetos que no sea dependiente de plataformas, modelos de desarrollo ni lenguajes de programación. Por todos estos motivos surge el concepto de SOAP (Simple Object Access Protocol).

5.1 Concepto de SOAP

La funcionalidad que aporta SOAP es la de proporcionar un mecanismo simple y ligero de intercambio de información entre dos puntos usando el lenguaje XML. SOAP no es más que un mecanismo sencillo de expresar la información mediante un modelo de empaquetado de datos modular y una serie de mecanismos de codificación de datos. Esto permite que SOAP sea utilizado en un amplio rango de servidores de aplicaciones que trabajen mediante el modelo de comunicación RPC (Remote Procedure Call).

SOAP consta de tres partes:

- El SOAP envelope que define el marco de trabajo que determina qué se puede introducir en un mensaje, quién debería hacerlo y si esa operación es opcional u obligatoria.
- Las reglas de codificación SOAP que definen el mecanismo de serialización que será usado para encapsular en los mensajes los distintos tipos de datos.
- La representación SOAP RPC que define un modo de funcionamiento a la hora de realizar llamadas a procedimientos remotos y la obtención de sus resultados.

5.2 Objetivos de SOAP

A la hora de realizar el diseño de SOAP se han tenido en cuenta una serie de consideraciones con el fin de cumplir una serie de objetivos claros, objetivos que le darán el potencial que reside en SOAP y que le harán tan atractivo.

Estos objetivos son:

1) Establecer un protocolo estándar de invocación a servicios remotos que esté basado en protocolos estándares de uso frecuente en Internet, como son HTTP (Hiper Text Transport Protocol) para la transmisión y XML (eXtensible Markup Language) para la codificación de los datos.

2) Independencia de plataforma hardware, lenguaje de programación e implementación del servicio Web.

El logro de estos objetivos ha hecho de SOAP un protocolo extremadamente útil, ya que el protocolo de comunicación HTTP es el empleado para la conexión sobre Internet, por lo que se garantiza que cualquier cliente con un navegador estándar pueda conectarse con un servidor remoto. Además, los datos en la transmisión se empaquetan o serializan con el lenguaje XML, que se ha convertido en algo imprescindible en el intercambio de datos ya que es capaz de salvar las incompatibilidades que existían en el resto de protocolos de representación de datos de la red.

Por otra parte, los servidores Web pueden procesar las peticiones de usuario empleando tecnologías tales como Servlets, páginas ASP (Active Server Pages), páginas JSP (Java Server Pages) o sencillamente un servidor de aplicaciones con invocación de objetos mediante CORBA, COM o EJB.

La especificación SOAP (SOAP Specification 1.1) indica que las aplicaciones deben ser independientes del lenguaje de desarrollo, por lo que las aplicaciones cliente y servidor pueden estar escritas con HTML, DHTML, Java, Visual Basic o cualquier otra herramienta o lenguaje disponibles.

5.3 Ventajas del protocolo SOAP

A continuación se lista cuáles son las principales ventajas de hacer uso de este protocolo:

No está asociado con ningún lenguaje: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista pero los desarrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que

utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como en Java, y la plataforma como Microsoft .Net.

No se encuentra fuertemente asociado a ningún protocolo de transporte: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.

No está atado a ninguna infraestructura de objeto distribuido La mayoría de los sistemas de objetos distribuidos se pueden extender, y ya lo están alguno de ellos para que admitan SOAP.

Aprovecha los estándares existentes en la industria: Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se ha mencionado SOAP no define un medio de trasporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.

Permite la interoperabilidad entre múltiples entornos: SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas. Por ejemplo, una aplicación de escritorio que se ejecute en una PC puede comunicarse con una aplicación del back-end ejecutándose en un mainframe capaz de enviar y recibir XML sobre HTTP.

5.4 Partes de un mensaje SOAP

Un mensaje SOAP no es más que un documento en formato XML que está constituido por tres partes bien definidas que son: el SOAP envelope, el SOAP header de carácter opcional y el SOAP body. Cada uno de estos elementos contiene lo siguiente:

El **envelope** es el elemento más importante y de mayor jerarquía dentro del documento XML y representa al mensaje que lleva almacenado dicho documento.

El **header** es un mecanismo genérico que se utiliza para añadir características adicionales al mensaje SOAP. El modo en la que se añadan cada uno de los campos dependerá exclusivamente del servicio implementado entre cliente y servidor, de forma que cliente y servidor deberán estar de acuerdo con la jerarquía con la que se hayan añadido los distintos campos. De esta forma será sencillo separar entre sí los distintos datos a transmitir dentro del mensaje.

El **body** es un contenedor de información en el cual se almacenarán los datos que se quieran transmitir de lado a lado de la comunicación. Dentro de este campo, SOAP define un elemento de uso opcional denominado Fault utilizado en los mensajes de respuesta para indicar al cliente algún error ocurrido en el servidor.

Un ejemplo de uso del header, donde se indica un cierto parámetro útil para el servicio Web que le indica cómo debe procesar el mensaje sería:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP- ENV:encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/">

    <SOAP-ENV:Header>
        <t:Transaction  xmlns:t="some-URI"  SOAP-ENV:mustUnderstand="1">
            5
        </t:Transaction>
    </SOAP-ENV:Header>

    <SOAP-ENV:Body>
        <getQuote xmlns= "http://namespaces.cafeconleche.org/xmljava/ch2/">
            <symbol>RHAT</symbol>
        </getQuote>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Un ejemplo de uso del nuevo elemento body sería el siguiente, en el que se ha detectado un error en la aplicación que corre sobre el servidor que provoca que no opere convenientemente, por lo que se le indica al cliente:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/">

    <SOAP-ENV:Body>

        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Server</faultcode>

            <faultstring>Server Error</faultstring>

        </SOAP-ENV:Fault>

    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

El atributo encodingStyle se utiliza para indicar las reglas de serialización utilizadas en el mensaje SOAP. No existe un formato de codificación por defecto, sino que existen una serie de posibles formatos a utilizar. El valor de este atributo es una lista ordenada de una o más URIs que identifican la regla o reglas de serialización que pueden ser

utilizadas en el mensaje, en el orden en el que se han de aplicar. De entre todas las posibles, las más utilizadas son:

- http://schemas.xmlsoap.org/soap/encoding/
- http://my.host/encoding/restricted
- http://my.host/encoding/

Todo mensaje SOAP debe tener un elemento Envelope asociado a la dirección de red http://schemas.xmlsoap.org/soap/envelope/. Si un mensaje recibido por una aplicación SOAP contiene en este elemento un valor distinto al anterior la aplicación trataría dicho mensaje como erróneo.

Ejemplo de un mensaje SOAP

Un ejemplo sencillo de mensajes SOAP Supongamos que tenemos un servicio Web el cual contiene una relación de productos junto con una serie de características de éstos, y supongamos que le queremos hacer una consulta acerca del precio de uno de los productos cuyo código es RHAT, el código relativo a la petición de consulta en lenguaje SOAP sería:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
                xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
          SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

  <SOAP-ENV:Body>

      <getQuote xmlns="http://namespaces.cafeconleche.org/xmljava/ch2/">
            <symbol>RHAT</symbol>
      </getQuote>

  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Como respuesta a esta petición vendría de vuelta el siguiente mensaje en el que se le dice que el precio del producto pedido es 4.12:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
                xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
          SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

  <SOAP-ENV:Body>

       <Quote xmlns="http://namespaces.cafeconleche.org/xmljava/ch2/">
              <Price>4.12</Price>
       </Quote>

  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5.5 Protocolo HTTP mediante SOAP

Una de las funcionalidades con las que cuenta SOAP y que lo hace extremadamente atractivo es el envío de mensajes SOAP vía protocolo HTTP, todo ello realizado de forma directa por medio de las librerías de SOAP. Este hecho provoca que existen ciertas diferencias entre un mensaje SOAP normal y uno enviado haciendo uso del protocolo HTTP.

De esta forma deben cumplirse dos aspectos:

- El HTTP header del mensaje de petición al servicio Web debe contener un campo SOAPAction. El campo SOAPAction alerta a los servidores Web y firewalls por los que pase de que el mensaje contiene documento SOAP, esto facilita a dichos firewalls el filtrado de las peticiones SOAP sin necesidad de tener que explorar el contenido del body del mensaje.
- Si la petición SOAP al servicio Web falla, el servidor debe devolver en el mensaje de respuesta un error del tipo HTTP 500 Internal Server Error en vez de un 200 OK que se envía cuando todo ha ido bien.

WSDL

En este apartado se procederá a describir el funcionamiento y composición de WSDL (Lenguaje de Descripción de Servicios Web).

El lenguaje de descripción de servicios Web (WSDL, Web Service Description Language) es un dialecto basado en XML sobre el esquema que describe un servicio Web. Un documento WSDL proporciona la información necesaria al cliente para interaccionar con el servicio Web. WSDL es extensible y se pude utilizar para describir, prácticamente, cualquier servicio de red, incluyendo SOAP sobre HTTP e incluso protocolos que no se basan en XML como DCOM sobre UDP.

Dado que los protocolos de comunicaciones y los formatos de mensajes están estandarizados en la comunidad del Web, cada día aumenta la posibilidad e importancia de describir las comunicaciones de forma estructurada. WSDL afronta esta necesidad definiendo una gramática XML que describe los servicios de red como colecciones de puntos finales de comunicación capaces de intercambiar mensajes. Las definiciones de servicio de WSDL proporcionan documentación para sistemas distribuidos y sirven como fórmula para automatizar los detalles que toman parte en la comunicación entre aplicaciones.

Los documentos WSDL definen los servicios como colecciones de puntos finales de red o puertos. En WSDL, la definición abstracta de puntos finales y de mensajes se separa de la instalación concreta de red o de los enlaces del formato de datos. Esto permite la reutilización de definiciones abstractas: mensajes, que son descripciones abstractas de los datos que se están intercambiando y tipos de puertos, que son colecciones abstractas de operaciones. Las especificaciones concretas del protocolo y del formato de datos para un tipo de puerto determinado constituyen un enlace reutilizable.

Un puerto se define por la asociación de una dirección de red y un enlace reutilizable; una colección de puertos define un servicio.

Por esta razón, un documento WSDL utiliza los siguientes elementos en la definición de servicios de red:

- Types: contenedor de definiciones del tipo de datos que utiliza algún sistema de tipos (por ejemplo XSD).
- Message: definición abstracta y escrita de los datos que se están comunicando.
- Operation: descripción abstracta de una acción admitida por el servicio.
- Port Type: conjunto abstracto de operaciones admitidas por uno o más puntos finales.
- Binding: especificación del protocolo y del formato de datos para un tipo de puerto determinado.
- Port: punto final único que se define como la combinación de un enlace y una dirección de red.
- Service: colección de puntos finales relacionados.

5.6 Utilización de SOAP y WSDL para el servicio Web

Al momento de ejecutar el algoritmo del sistema de notificaciones, éste requiere de una comunicación con el servidor de Firebase para generar las notificaciones a todos los usuarios que estén subscriptos a un tópico en especial. Es por ello, que se requiere hacer uso de protocolo de comunicación SOAP en conjunto con WDSL.

Se envía un mensaje SOAP personalizado al servidor de Firebase, con los campos requeridos para llevar a cabo las notificaciones.

En el Head del mensaje, se debe especificar el token de autentificación con el que se va a validar los permisos que tiene el Web Service para realizar acciones sobre el servidor de Firebase, en conjunto con un Content-Type que determinará el formato en el cual son enviados los datos. Para este caso, se optó por usar json como formato estándar para estructurar la información.

Con lo que respecta al Body, éste almacenará la información para poder generar la notificación a los usuarios. Entre los campos que conforman al Body podemos encontrar:

- Title: Titulo de la notificación que se va a mostrar en los dispositivos móviles.
- Body: Mensaje informativo del cuerpo de la notificación describiendo el hecho de que una nueva persona se encuentra desaparecida.
- Click_Action: Acción que va a ejecutar cuando el usuario presione sobre la notificación. En este caso, sería mostrar toda la información referida a la persona desaparecida.
- To: Representa el tópico al cual será destinada la notificación.
- Persona: Representa la información de la persona que se encuentra actualmente desaparecida.

Una vez que el mensaje SOAP es construido, se envía dicho mensaje a un URL suministrado por Firebase, donde se acepta o rechaza su recepción. En caso de que el mensaje sea aceptado, el servidor de Firebase responderá con un mensaje de “success” y enviará la notificación a todos los usuarios suscriptos al tópico. Caso contrario, el servidor de Firebase responderá con “failure” y no se notificará a los usuarios subscriptos al tópico.

Vale aclarar que, se requiere de un envío de un mensaje SOAP hacia el servidor de Firebase por cada tópico a la que se desea enviar la notificación.

5.7 Conclusión

Es de suma importancia contar con herramientas sencillas para la comunicación entre servidores. Para este proyecto en particular, solo basta con diferentes comunicaciones sincrónica entre el Web Service del proyecto con el Servidor de Firebase, para generar todas las notificaciones. Es por ello, que se hace uso de SOAP en conjunto con WSDL.

Su sencillez en conjunto con su facilidad de uso e independencia del lenguaje, son algunas de la características más importantes que conllevan a elección de dicho protocolo.

Conclusión

Durante el desarrollo del informe, se evaluaron y fundamentaron las elecciones de las diferentes alternativas acerca de las tecnologías y los servicios externos. Dichos análisis, surgen de la necesidad de construir un sistema con un alto grado de calidad para poder ser utilizado a futuro.

Posterior a estos análisis, se procederá a la implementación final del sistema completo, respetando tanto dichas elecciones como la arquitectura adoptada.

Glosario

CMS: Component Management System.

IEEE: The Institute of Electrical and Electronics Engineers.

PHP: Pre Hypertext – Processor.

MVC: Model - View - Controller.

HTML: HyperText Markup Language.

SQL: Structured Query Language.

AJAX: Asynchronous JavaScript And XML.

ORM: Object-Relational Mapping.

DOM: Document Object Model.

SDK: Software Development Kit.

JSON: JavaScript Object Notation.

BFS: Breadth First Search.

SOAP: Simple Object Access Protocol.

XML: eXtensible Markup Language.

SMTP: Simple Mail Transfer Protocol.

HTTP: HyperText Transfer Protocol.

WSDL: Web Services Description Language.