REALIZADO POR	FECHA	FIRMA
Lautaro Sikh		
REVISADO POR	**FECHA**	**FIRMA**
Emmanuel Rojas Fredini		
REVISADO POR	**FECHA**	**FIRMA**
-	-	**-**
APROBADO POR	**FECHA**	**FIRMA**
Walter Schulte		

Nombre del Proyecto: DESARROLLO DE UNA APLICACIÓN EN ANDROID PARA UBICAR PUNTOS DE INTERÉS EN LAS DEPENDENCIAS DE LA UNL

Periodo del Informe: 21/08/2017 – 6/11/2017

Alcance: Etapas 1 y 2

Estado del Proyecto

Cronograma

Etapa 1: Análisis del problema e identificación de requerimientos

Actividad	Fecha realización	
	Estimada	Real
Investigación de la plataforma de desarrollo y lenguaje	21/08	4/09
Investigación de la API de Google Maps	28/08	8/09
Investigación acerca de Web Services	31/08	8/09
Recolección de requerimientos y elección de los mismos	7/09	23/09
Creación de documento de requerimientos	15/09	23/09

Resultados obtenidos

Se deja un anexo con un detalle de algunas de las actividades

Etapa 2: Diseño de la propuesta de solución

Actividad	Fecha realización	
	Estimada	Real
Elección de las tecnologías a utilizar	26/09	30/09
Definición de las funcionalidades	28/09	16/10
Diseño de la base de datos	3/10	16/10
Diseño del Web Service	5/10	21/10
Diseño de la aplicación	10/10	21/10
Diseño de integración de las partes	16/10	21/10
Redacción de la propuesta de solución	19/10	21/10

Resultados obtenidos

Se deja un anexo con un detalle de algunas de las actividades

Riesgos

Riesgo	Se efectivizó		Impacto	Mitigación
Falta de disponibilidad del recurso humano del proyecto	Sí	No	Retraso de inicio del proyecto y ejecución de algunas actividades	Re calendarizar actividades y recuperar horas los fines de semana.
Desperfectos en la computadora de trabajo y perdida de la Información	Sí	No		
Desperfectos en el dispositivo Android y no poder realizar pruebas de la aplicación	Sí	No		
	Sí	No		

Riesgos futuros	Probabilidad	Impacto
Falta de disponibilidad del recurso humano del proyecto	Alta	Medio

Notas

Tomé un puesto de trabajo fijo de 8hs diarias, por lo que es muy probable que este riesgo se efectivice durante todo el proyecto.

ANEXO

Investigación de la plataforma de desarrollo y lenguaje

En esta actividad se investigaron las posibles herramientas de trabajo y el lenguaje de programación a utilizar. El desarrollo del Web Service se puede llevar a cabo con la herramienta *eclipse*, que es uno de los Entornos de Desarrollo Integrado (IDE en ingles) más utilizados para desarrollo de aplicaciones bajo lenguaje Java. Éste será el lenguaje de programación a utilizar porque soporta todas las necesidades del proyecto y mantiene la homogeneidad con Android ya que su *Software Development Kit* (SDK) está basado en Java. Además, hoy en día existen numerosos frameworks desarrollados bajo este lenguaje que pueden ser muy provechosos para el proyecto, cómo son Hibernate o Spring y cuya inclusión podría brindar muchas facilidades.

La aplicación para dispositivos móviles será desarrollada con Android Studio porque es el IDE de desarrollo oficial para esta tecnología. Ambas herramientas mencionadas son de uso gratuito y poseen una gran documentación en Internet y comunidades muy activas, lo que es un punto muy fuerte a la hora de escogerlas.

El gestor de bases de datos a utilizar será MySQL ya que es una base de datos open source y es de las más populares en la actualidad, que igualmente cuenta con basta documentación y comunidad activa. No es el único, existen diversas herramientas similares pero MySQL es lo suficientemente completa para las necesidades del proyecto. También existen diversos drivers para efectuar conexiones a la misma, por lo tanto este tema no será un problema.

Cómo se menciono antes, Hibernate es un framework para mapeos objetos-relaciones para Java y .NET, que simplifica la transición de llevar atributos persistidos en una base de datos, a un objeto perteneciente al modelo de objetos una aplicación. Pero también provee muchas facilidades para realizar consultas a dicha base de datos y crear objetos en función de ellos y ejecutar todo tipo de sentencias. Este *Object-Relational mapping* (ORM) posee soporte para conexiones a bases de datos MySQL lo que permite la fácil integración de esta tecnología.

En cuanto al desarrollo de la aplicación bajo tecnología Android, la misma será compatible con versiones iguales o superiores a Android 4.4 debido a las prestaciones

que brinda esta versión teniendo en cuenta el alto porcentaje de activaciones de versiones incluso superiores a esta, que es del 90% de los dispositivos según las estadísticas que brinda Android Studio. Podrían considerarse versiones superiores llegado al caso.

Investigación de la API de Google Maps

La API de Google Maps para desarrollo bajo tecnología Android permite al desarrollador colocar un mapa de Google Maps en la pantalla de la aplicación e interactuar con él tal como si se estuviera utilizando en una computadora. Para poder acceder a esta herramienta se debe contar con una *key* provista por Google. Esta, es una cadena de caracteres alfanuméricos, mayúsculas y minúsculas, que se proveen a un determinado desarrollador para incluirla en sus productos.

Las mismas se pueden obtener de la web de la API de Google Maps para desarrolladores.[1]

Las *keys* son de uso gratuito siempre y cuando la aplicación desarrollada no posea fines lucrativos, por lo tanto en principio no habría problemas en utilizar una para los fines del proyecto, que es académico.

La API provee de muchas facilidades que son útiles para el proyecto. Permite interactuar con un mapa, dejando que el usuario se desplace hacia cualquier dirección e incluso hacer *zoom in* y *zoom out*. También permite incorporar marcadores de tipo pines con mensajes o imágenes que servirán para indicar la posición del punto de interés buscado o la del usuario.

Otra facilidad es la posibilidad de incorporar *Ground Overlays*, que son imágenes que se ubican por encima del mapa. En nuestro caso serán los planos de los edificios por donde pase el camino que el usuario deba recorrer.

Dicho camino se puede trazar con otra herramienta que son las polilineas. Las mismas son lineas que van de marcador en marcador por encima del mapa, como se suele ver en *Google Maps* cuando se pide información de como llegar a tal punto desde otro.

1 https://developers.google.com/maps/documentation/android-api

El mapa, las polilíneas, marcador y *overlays* serán en principio las 4 herramientas principales para llevar a cabo el proyecto, pudiéndose incluir otras según se crea necesario en un futuro para implementar funcionalidades que puedan surgir, como ser marcadores planos que se mueven de forma animada simulando la posición del usuario o el sensor de rotación del teléfono para rotar el mapa apuntando al punto cardinal que el usuario este viendo.

Investigación acerca de Web Services

El intercambio de datos entre la aplicación y la base de datos se hará mediante un Web Service. Se decidió que la arquitectura del mismo sea *Representational State Transfer* (REST) . Esto se debe a que REST se ajusta a las necesidades del proyecto y es de fácil y rápida implementación.

La disyuntiva planteada en este aspecto era sobre una representación REST o SOAP (*Simple Object Access Protocol)*. Por un lado, la arquitectura REST es más nueva y de diseño más sencillo. Aunque no ampara en cuestiones de seguridad cómo si lo hace SOAP, se determino que el intercambio de datos entre cliente-servidor no posee contenido sensible y por lo tanto no es necesario realizar esfuerzos extra en preservar la seguridad de la comunicación.

Él mismo está basado en el protocolo de comunicación HTTP que brinda una serie de primitivas básicas de petición e intercambio de datos, de las cuales la más importante para este proyecto es la petición GET, que a partir (o no) de ciertos parámetros, realiza consultas en el lado servidor y las envía de nuevo por el mismo canal. Cada mensaje HTTP contiene toda la información necesaria para ejecutar la petición por lo tanto no se requiere guardar ningún tipo de estado de la comunicación sino que todas son independientes entre sí. La mayoría de las API de dominio público de los servicios que usamos habitualmente están constituidos en REST (Facebook, Yahoo, Twitter, Mercado Libre, entre otras).

La aplicación no realiza sentencias que persistan cambios en la base de datos, sólo consume información de la misma por lo tanto se determinó que no hace falta tener una arquitectura con tantos recursos, sino que es más conveniente contar con una arquitectura simple y robusta.

Además, la tendencia indica que la arquitectura REST será la más utilizada en el futuro. Todas las decisiones de implementación se tomaron teniendo en cuenta el grado de impacto de la herramienta en el mercado y su uso en el ámbito global.

La información obtenida de la base de datos será enviada en formato *JavaScript Object Notation* (JSON) a la aplicación. Se escogió el formato JSON ya que a diferencia de *Extensible Markup Language* (XML), posee mayor soporte y es de más fácil lectura a simple vista. No requiere de etiquetas cómo el caso de XML por lo que reduce el ancho de banda de los datos a enviar, lo cual es un punto a favor y al igual que REST la tendencia entre JSON y XML se inclina a su favor.

Recolección de requerimientos, elección de los mismos y funcionalidades básicas

En primera instancia, se pretenden desarrollar una aplicación y un Web Service que cumplan con los siguientes requerimientos:

- Qué el usuario pueda elegir entre las categorías de búsqueda: Aulas, Oficinas, Baños, Bares (Cantinas).
- Qué el usuario pueda elegir el edificio (dependencia) en donde está buscando.
- Qué el Web Service responda la petición del usuario y la aplicación muestre las opciones (por ejemplo, Oficinas de FICH podría recibir como respuesta: Alumnado, Bedelía, Secretaría Académica, etc)
- Qué el usuario seleccione un punto de interés y se envié una petición al Web Service para obtener el conjunto de marcadores que definen la/las polilínea/s a recorrer.
- La aplicación deberá ser capaz de geoposicionar al usuario utilizando los sensores del móvil.
- La aplicación mostrará imágenes de algunos de los puntos de interés.
- La aplicación brindara información adicional sobre las unidades académicas y dependencias, cómo ser datos de contacto o ubicación.
- La aplicación deberá poder leer códigos QR con la latitud, longitud y piso en el que se encuentra el usuario para facilitar el geoposicionamiento.
- La aplicación podrá sincronizar con el Web Service para determinar si los planos que están almacenados en el móvil están actualizados. De caso contrario, podrá descargar los nuevos.

También se definieron algunas funcionalidades extra que podrían ser de interés y se evaluara su desarrollo, cómo ser tenerla posibilidad de guardar búsquedas para consultarlas en otro momento, contar con un modo de visualización de elementos de seguridad, por ejemplo salidas de emergencia, matafuegos, etc o contar con un módulo con información general de las dependencias de la UNL (nombre, dirección, teléfono, ubicación en el mapa, entre otras).

Diseño de la Base de Datos

El diseño de la base de datos se ha previsto de forma relacional. A continuación se deja el mismo con un detalle de cada tabla.

Se deja el diagrama para visualizar. El mismo es una aproximación inicial y podría sufrir cambios en función de nuevas necesidades o requerimientos técnicos.

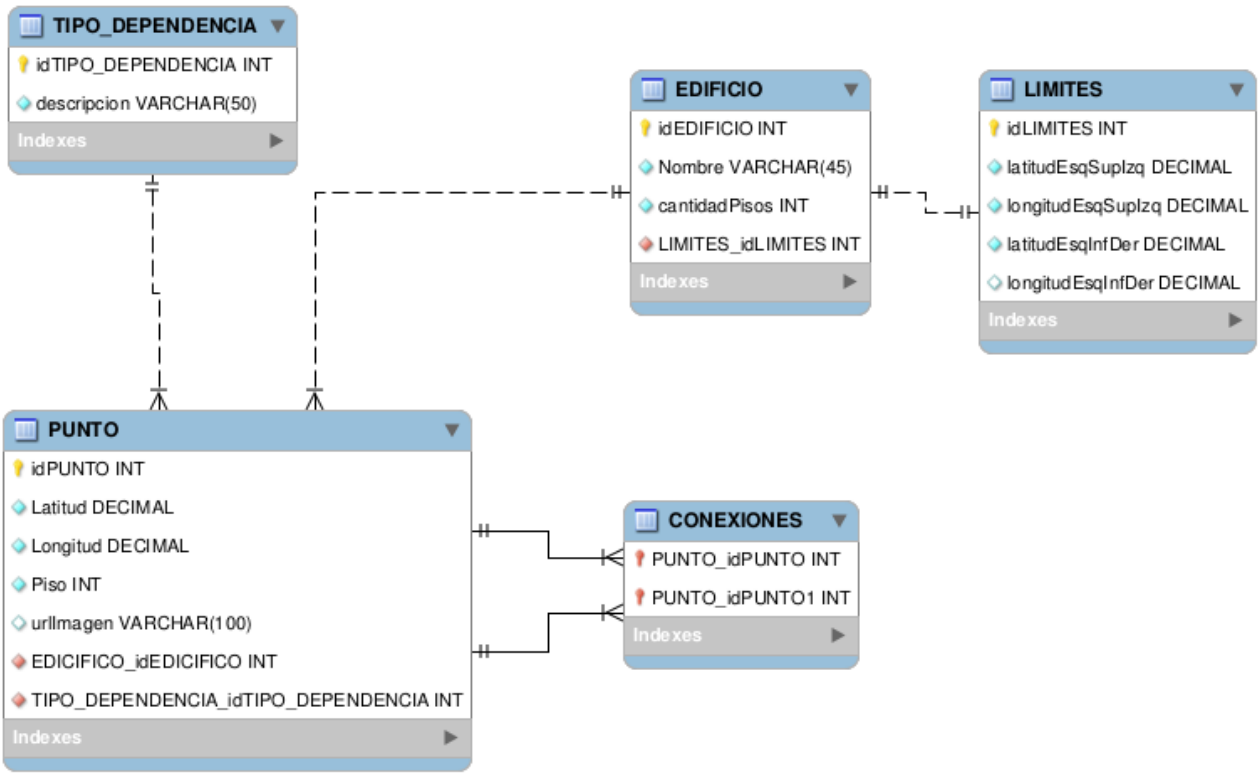

Ilustración 1: Diseño de la base de datos

PUNTO: esta tabla representa a las entidades que serán los puntos de interés en la Universidad. Cada punto posee una latitud, longitud y número de piso en el que se

encuentran. También poseen la dirección URL dentro del servidor en donde pueden haber o no, una imagen de ese punto para mostrar en la aplicación.

EDIFICIO: representa a cada edificio de la universidad. Se cargará su nombre y la cantidad de pisos que tiene dicho edificio.

TIPO_DEPENDENCIA: representa el tipo de dependencia que se va a relevar. En una primera instancia los posibles valores del campo descripción de esta tabla podrían ser: “Aula”, “Oficina”, “Baño”, “Bar” y un registro más para aquellos puntos que no representan dependencias sino que son solo para marcar el camino del grafo.

LIMITES: registro totalmente dependiente de la entidad EDIFICIO para poder determinar su posición en el mapa a partir de la longitud y latitud de sus esquinas superiores e inferiores.

CONEXIONES: tabla que se genera de una relación *Many To Many* de la entidad PUNTO. Cada punto puede tener una cantidad indefinida de vecinos, que también son puntos, y dicho punto puede ser vecino de otra cantidad indefinida de puntos.

Diseño del Web Service

Para el diseño del Web Service se propone realizar un modelo de 3 capas.

La primera de ellas es la capa de aplicación que atenderá las *requests* que lleguen al Web Service. Esta se comunicará con una capa de servicio que poseerá las clases que ejecuten la lógica de negocio y tendrá acceso a la tercer capa, la de acceso a datos. Esta última será la encargada de todo lo referente a transacciones en la base de datos.

Un diseño a modo de ejemplo se deja a continuación. El mismo es tentativo y podrá ser complementado conforme al avance del proyecto en función de las necesidades que vayan surgiendo.

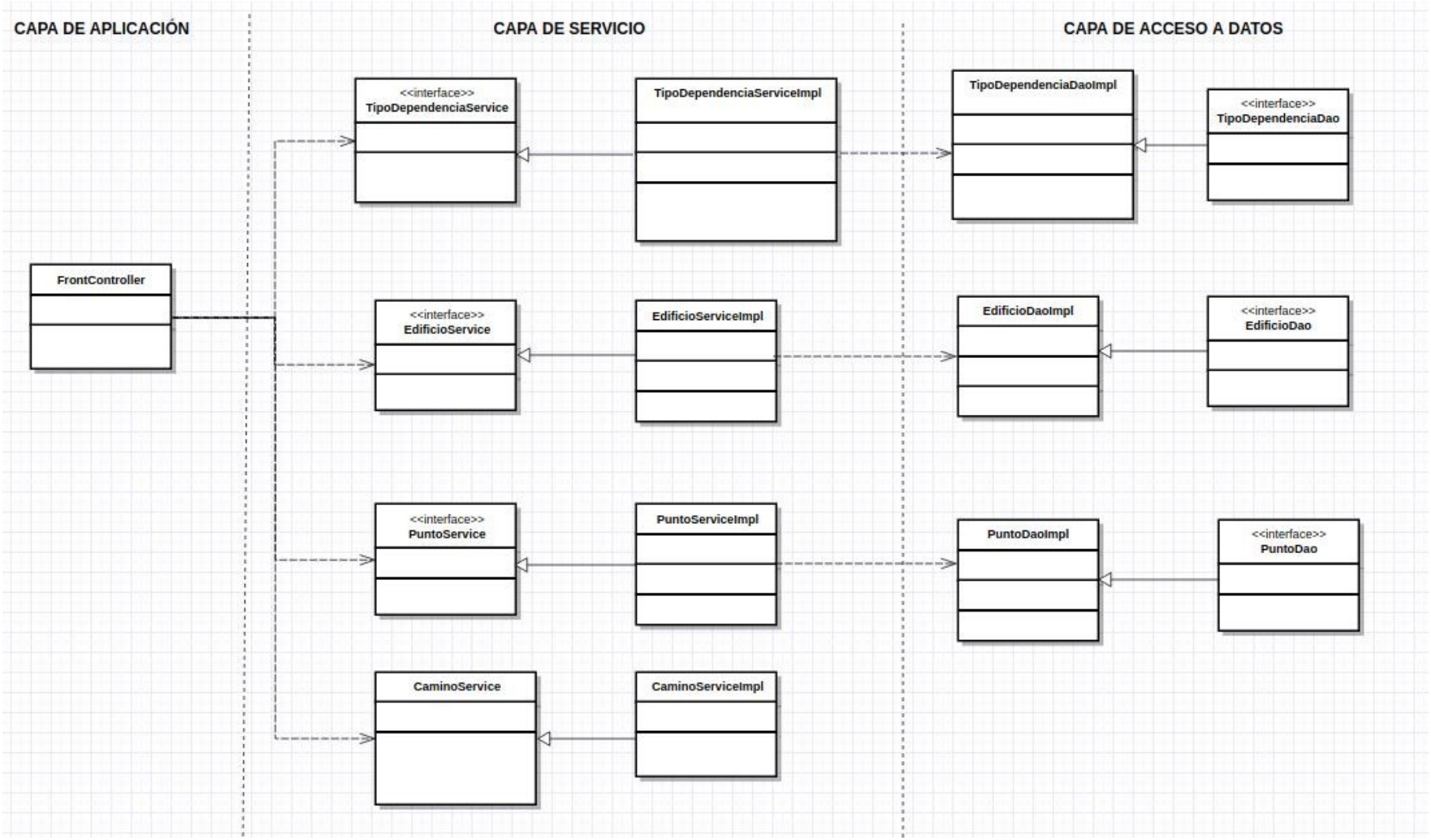

Ilustración 2: Diseño del Web Service

Aquí, por ejemplo, el controlador frontal podría recibir una petición del usuario para visualizar todas las aulas del edificio FICH. Un método en esta clase encargará de comunicarle a la implementación de servicio de la entidad Edificio que debe recuperar todas aquellas entidades que cumplan con la condición de que TipoDependencia sea igual a Aula y el edificio sea la FICH. Esta regla de negocio es procesada y pasada a la clase de acceso a datos correspondiente para recuperar las entidades correspondientes y ser devuelto al controlador frontal.

También se estima que exista otro módulo que ejecute la lógica de negocio referida la ejecución del algoritmo de búsqueda así como también otros módulos que realicen tareas a fin de cumplir con el cometido. Los mismos integraran la capa de servicios y todos aquellos que mantengan relación directa con la base de datos estarán en la capa de acceso a datos.

Integración de partes y propuesta de solución

El flujo principal de actividades que se propone para brindar una solución al problema planteado es el siguiente:

- El usuario abre la aplicación. La misma constata mediante el Web Service que los planos que posee en su base de datos son los mismo que en el lado servidor. En caso contrario, descarga las nuevas versiones.

- El usuario, con el sensor de ubicación prendido y conexión a Internet, elige de la aplicación que tipo de dependencia está buscando y selecciona el edificio de la misma.

- La aplicación envía una petición GET al Web Service que contendrá la información que seleccionó el usuario.

- El Web Service atiende esta petición, y a partir de las clases de servicio, recupera una lista de las dependencias a devolver a la aplicación, previamente convirtiendo esta información en un JSON. Por ejemplo, si el usuario selecciono Aulas de FICH, el resultado que recibirá del Web Service será una lista que contendrá las claves: “Aula 1”, “Aula 2”, etc.

- Ahora el usuario selecciona una de estas opciones y oprime un botón que ejecute la búsqueda.

- Se envía otra petición del tipo GET al Web Service que contendrá: dependencia a ubicar (ej: “Aula 3 – FICH”), latitud y longitud del usuario y el piso donde se encuentre si este no fuera planta baja.

- El Web Service recibe esta información y ubica al usuario a partir de su posición para saber en que parte de la Universidad está.

- Recupera las entidades Punto pertinentes y ejecuta un algoritmo de búsqueda de costo uniforme, ya que se pretende encontrar el camino más corto entre el usuario y el destino, hasta encontrar en el grafo la opción seleccionada por el usuario.

- Se genera un JSON con la información de todos los puntos por donde debe pasar el usuario hasta llegar al objetivo y se envía.

- La aplicación toma esta información y le vuelca en Google Maps. La sucesión de puntos constituyen polilineas y los marcadores indican la posición inicial y final del usuario.

- Si el punto objetivo estuviera en un piso distinto al que se encuentra el usuario, este podrá ir cambiando de plantas y visualizar cada una de ellas.