ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"APLICACIÓN DE ESPOL PARA WINDOWS Y WINDOWS MOBILE"

INFORME DE PROYECTO INTEGRADOR

Previa a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES
ORIENTACIÓN SISTEMAS DE INFORMACIÓN

JUAN PABLO CAAMAÑO ESPINOZA

GUAYAQUIL – ECUADOR

AÑO: 2015

AGRADECIMIENTOS

Mis más sinceros agradecimientos a Dios, mi familia, mis amigos y cada una de las personas que han contribuido a mi formación personal y profesional.

DEDICATORIA

El presente proyecto lo dedico a cada una de las personas que se esmeran por construir un mundo mejor, que ponen el bienestar común por encima del bienestar propio. También a cada una de esas personas que todos los días ponen su grano de arena para hacer de la ESPOL la mejor universidad del Ecuador.

TRIBUNAL DE EVALUACIÓN

Ing. Guido Caicedo Rossi

PROFESOR EVALUADOR

Ing. Rafael Bonilla Armijos

PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me corresponde exclusivamente; y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Juan Caamaño Espinoza

RESUMEN

La Escuela Superior Politécnica de Litoral, ESPOL, tiene actualmente un problema con la capacidad para movilizar a sus estudiantes, debido a que los buses de TransEspol no se dan abasto para satisfacer la demanda existente, por lo que se necesita implementar un mecanismo complementario de transporte. Una forma de ayudar a la movilización de los estudiantes es incentivar a que aquellos que poseen un vehículo lleven a los que no poseen.

Por medio de una aplicación se busca que los estudiantes conozcan mejor las rutas disponibles para optimizar su uso, además por medio de la aplicación fomentar la movilización entre estudiantes, es decir que estudiantes que tengan lugares disponibles en sus vehículos los oferten a estudiantes que deseen llegar al campus de la ESPOL. El patrón usado para el desarrollo de la aplicación es Model View ViewModel, que tiene entre sus principales beneficios, la facilidad para ampliar a otras plataformas las aplicaciones desarrolladas utilizando MVVM.

ÍNDICE GENERAL

CAPÍTULO 1

1. ANÁLISIS DEL PROBLEMA.

La Escuela Superior Politécnica del Litoral, ESPOL, es una entidad de educación superior estatal de categoría A, donde hasta finales 2014 tenía matriculados 9.690 estudiantes [1, p. 40], su campus principal está en la Prosperina, en las periferias de la ciudad de Guayaquil. El inicio de la jornada de clases es a las 07:30.

El ingreso a la universidad se realiza por medio del transporte público (buses) con varias rutas desde distintos puntos de la ciudad, otros estudiantes ingresan al campus usando sus vehículos propios.

El problema es que, incluso existiendo varias alternativas de transporte proporcionadas, tanto por ESPOL como por actores privados, se tiene todavía un sector de estudiantes que no pueden ser atendidos por los servicios de transporte anteriormente mencionados.

1.1 Causas

Actualmente las rutas de transportes convencionales no satisfacen las necesidades crecientes en número de estudiantes que entran en horarios matutinos o salen por las tardes, por lo que a pesar de que los buses de varias rutas salen a capacidad completa, aún quedan estudiantes esperando en la parada por otro bus.

1.2 Efectos

El principal efecto de los problemas de transporte se refleja con la insatisfacción de los usuarios, quienes gastan mucho tiempo en movilizarse entre las ESPOL y sus domicilios

1.3 Soluciones similares

Empresas como Uber o EsayTaxi, promueven una alternativa a los taxis convencionales, mediante la cual los usuarios usando una aplicación pueden buscar un chofer que esté cerca de su ubicación geográfica para realizar el servicio de transporte.

Uber es una solución que comercializa el servicio de alquiler de choferes, por medio de su aplicación detecta la ubicación actual del usuario y envía al chofer más cercano, el cobro de la aplicación se hace mediante tarjeta de crédito, la cual debe ser registrada junto con los datos personales del usuario antes de hacer uso de la aplicación. Si algún chofer desea ingresar a proveer servicios en Uber debe cumplir una serie de requisitos que la empresa pone, una vez cumplidos todos, el conductor recibirá un teléfono para facilitar el servicio ofertado [2]. Uber no es un servicio de taxi compartido, la empresa utiliza vehículos particulares además de la opción de escoger el tipo de vehículo que deseas utilizar dependiendo del país en el que el usuario se encuentre. Uber es una buena alternativa cuando no se cuenta con dinero en efectivo para realizar el pago del servicio de transporte [2].

EasyTaxi acerca a los conductores de taxis tradicionales con los usuarios por medio de su aplicación. El usuario descarga la aplicación y se registra con sus datos y sus métodos de pago. Para hacer uso de la aplicación, por medio del GPS se detectará la ubicación actual, se puede editar la ubicación y posterior a esto confirmarla, se debe seleccionar la forma de pago y se procede a solicitar el servicio [3]. El chofer del taxi es quien decide si desea aceptar la solicitud, y proceder a brindar el servicio. Existen dos versiones de la aplicación una para usuarios y otra para ofertantes, para convertirse en ofertante de EasyTaxi se necesita ingresar una solicitud con varios documentos personales, realizar algunas pruebas entre ellas pruebas psicológicas y verificaciones domiciliarias al aspirante a ofertar el servicio [3].

CAPÍTULO 2

2. ANÁLISIS DE LA SOLUCIÓN.

El objetivo principal de la aplicación es proveer de una solución que complemente a los métodos de transportes que existen actualmente en la ESPOL (Figura 2.1).

Figura 2.1: Concepto de la pantalla principal de la aplicación

2.1 Descripción

Por medio de la aplicación, el estudiante podrá conocer de forma detallada las rutas de transporte, tanto de ida como de regreso hacia la ESPOL; con esto se busca una optimización de las rutas utilizadas, reduciendo la posibilidad de uso de la ruta principal, y optimizando el uso de las rutas alternas que tienen una baja demanda.

Para acceder a la aplicación, los usuarios primero deben iniciar sesión con sus credenciales de ESPOL, entonces podrán acceder a una pantalla donde tienen las distintas rutas de transporte (Figura 2.2). Seleccionando la ruta deseada, se puede visualizar el detalle de las paradas y los horarios de la ruta (Figura 2.3).

Agregando a los descrito anteriormente, la aplicación será el medio de generación y utilización de transporte entre estudiantes. Para describir con más exactitud esta función usaremos de ejemplo: un estudiante que todos los días sale desde la Ciudadela La Saiba al sur de Guayaquil a las 07:30 hacia la ESPOL

y tiene tres asientos disponibles, por medio de la aplicación, registra su vehículo, registra también su punto de partida o un punto de partida que estén en su recorrido diario (Figura 2.4).

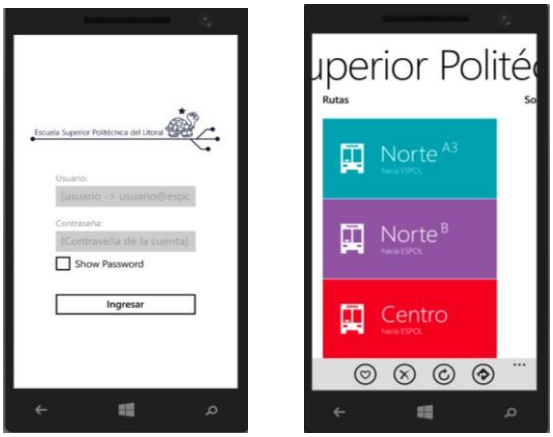

Figura 2.2: Inicio de sesión y menú de transportes en aplicación móvil

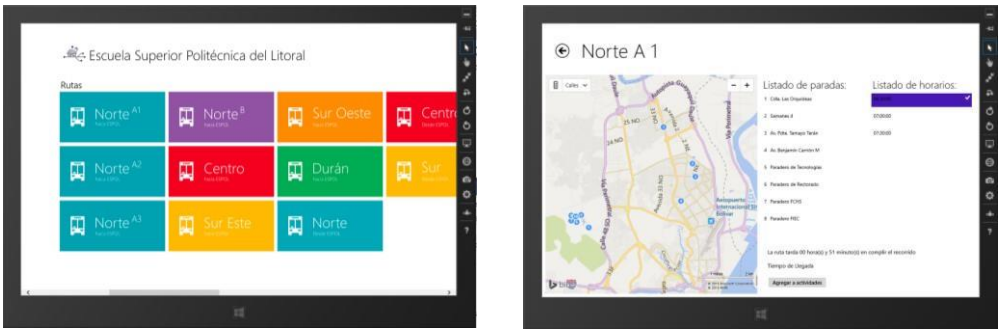

Figura 2.3: Menú de transporte y detalle de transporte en aplicación

A través de estos datos previamente ingresados, genera una oferta de transporte, colocando un nombre referencial de la oferta, la fecha y hora, el vehículo que va a utilizar, el punto de partida y el paradero de la ESPOL al que va a llegar; después de esto publica la oferta (Figura 2.5).

El solicitante del servicio tiene una ventana de búsqueda, donde puede ingresar la ubicación en un mapa y realizar la búsqueda de ofertantes de servicios en base a la mencionada ubicación.

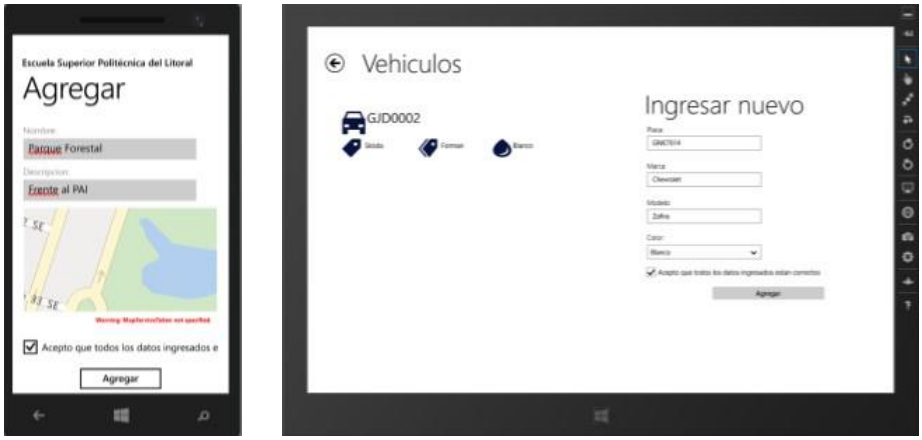

Figura 2.4: Ventana para agregar ubicación en aplicación móvil y ventana para agregar vehículo en aplicación

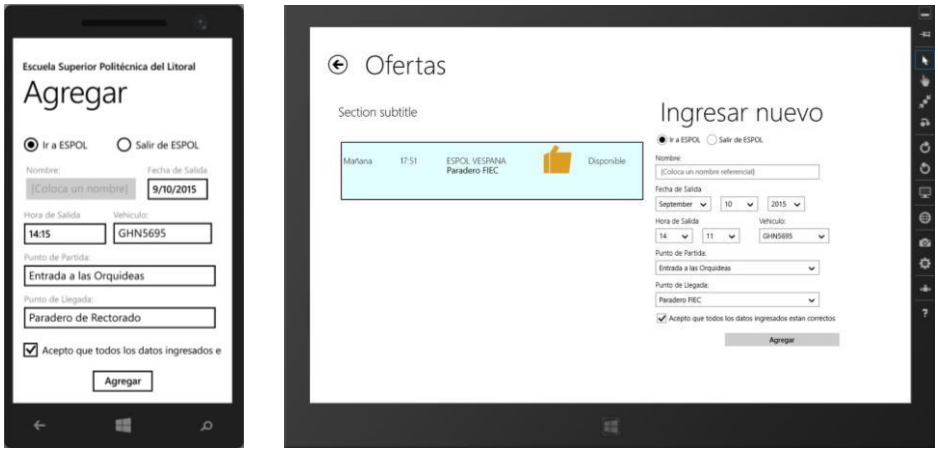

Figura 2.5: Ventana de ingreso de Ofertas de transporte en aplicación móvil y aplicación

Además a lo descrito la ventana de búsqueda tiene parámetros como fecha y hora, que pueden ayudar a enfocar la búsqueda de manera más minuciosa. Una vez ingresados los parámetros de búsqueda se mostrarán los resultados de la

misma, y el solicitante puede seleccionar la que mejor le convenga y enviar una solicitud de servicio al ofertante (Figura 2.6).

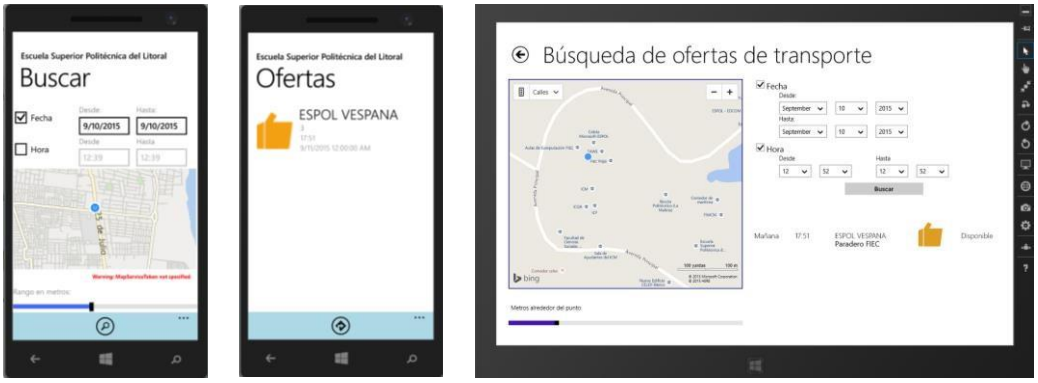

Figura 2.6: Ventana de búsqueda de ofertas en aplicación móvil y aplicación.

El ofertante puede visualizar en su pantalla de “Aprobar Solicitudes” el requerimiento enviado, y aprobar o negar la solicitud. Una vez realizada la acción, el solicitante puede verificar el estado de su solicitud y saber si ya puede acceder al servicio (Figura 2.7).

Si la solicitud fue aprobada, el solicitante puede ver datos adicionales del ofertante como datos del vehículo y dirección de correo electrónico, para de ser necesario establecer comunicación por correo electrónico con el ofertante (Figura 2.8).

Si el caso fuera que un estudiante desea salir de la ESPOL y brindar el servicio hacia una ubicación, puede de igual manera hacerlo, con los datos de ubicación y vehículo que fueron almacenados previamente, siguiendo el procedimiento anteriormente descrito

Otro ejemplo de como la aplicación puede ayudar considerablemente, es que entre dos o cuatro estudiantes que ofertan servicios en momentos y puntos parecidos, se turnen para ocupar un solo vehículo y no los 4 al mismo tiempo.

El uso de la aplicación tendrá ciertas restricciones para asegurar la calidad, seguridad y buen desempeño del mismo, entre las cuales tenemos:

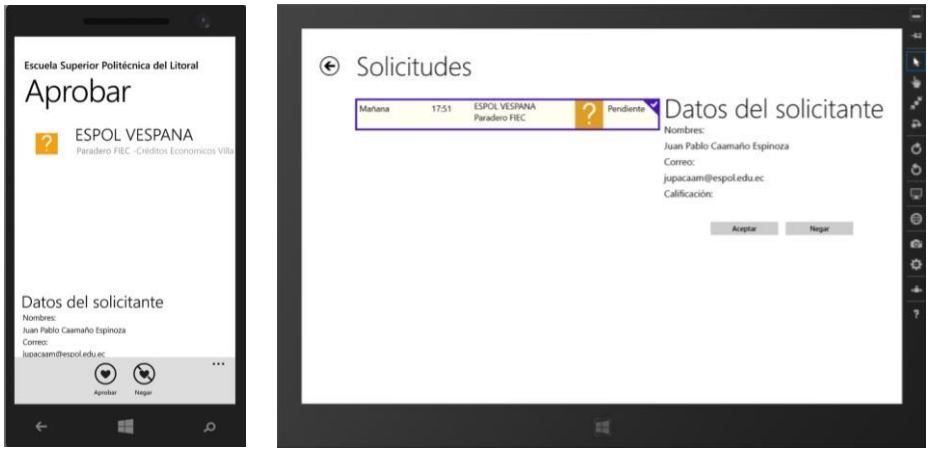

Figura 2.7: Ventana para aprobar solicitudes en aplicación móvil y aplicación.

Figura 2.8: Sección donde se visualiza las solicitudes aprobadas en aplicación móvil y aplicación.

- El servicio de transporte comunitario sólo estará disponible para estudiantes y ofertado por estudiantes.
- Para que un vehículo pueda ser usado para ofertar el servicio de transporte comunitario, debe pasar por una revisión y normativas dispuestas por el departamento de seguridad de la ESPOL.
- Un estudiante no puede registrar más de dos vehículos.
- Un estudiante no puede llevar a más de tres estudiantes.

- Un estudiante no pude registrar más de diez ubicaciones.
- Si el estudiante parte hacia la ESPOL, sus puntos de llegada deben ser alguno de los paraderos oficiales de la ESPOL, y si se dirige desde la ESPOL hacia una ubicación, debe de salir de las paradas oficiales de la ESPOL.

2.2 Resultados Esperados

Para una primera etapa, suponiendo que un promedio 400 autos ingresan a la ESPOL todos los días el siguiente semestre, si al menos un 10% de esos automóviles ofrecieran el servicio llevando como mínimo a 2 personas, tendríamos una cobertura estimada de 80 estudiantes, que daría paso a: tener menos estudiantes haciendo fila en la ruta principal hacia la ESPOL, tener menos estudiantes pidiendo transporte a la entrada de la ESPOL, y tener menos estudiantes yendo incómodos en las rutas alternas.

CAPÍTULO 3

3. IMPLEMENTACIÓN.

3.1 Arquitectura

La arquitectura que usaremos en el desarrollo de la aplicación es la arquitectura cliente servidor. Usaremos un servidor de datos de la ESPOL para almacenar la información de la aplicación, y los clientes serán teléfonos inteligentes y computadores personales.

3.2 Modelo de Desarrollo

Para el desarrollo de la App, nos basaremos en el modelo de desarrollo de App Universales que promueve Microsoft, este modelo nos permite codificar una sola vez y llegar a varias familias de dispositivos Microsoft en primera instancia. Microsoft, para hacer realidad el alcance ofrecido en “Universal Apps”, recomienda el uso del patrón de diseño Model View ViewModel.

3.1.1 Model View ViewModel

Model View ViewModel es un patrón muy popular entre los desarrolladores que usan XAML, derivado del “Presentation Model pattern” de Martin Fawler [4, p. 32], resalta muchas de las características que tiene el desarrollo para Windows, sobre todo el desarrollo de interfaces, para realizar una codificación limpia. La popularidad del patrón se ha extendido tanto que existe un framework open-source a disposición de programadores web [4, p. 32].

La principal misión de MVVM es crear una separación de objetivos, entre los datos que la aplicación necesita manipular y las vistas que se van a presentar a los usuarios. Sin esta separación, se podría crear clases enredadas que por un lado se preocupen de la lógica para el intercambio de datos, y al mismo tiempo la lógica para presentar datos al usuario [5, p. 25].

En base al alcance y las interacciones de la aplicación con el usuario, se generan las clases que se van a utilizar en la aplicación. La letra M del

modelo está relacionada a las clases que usa la aplicación, no tiene relación alguna con los servicios o la base de datos, pero es la representación de los datos que la aplicación debe utilizar en forma de clases. [4, p. 32].

Usando las herramientas de desarrollo de Microsoft para utilizar el patrón MVVM, existen dos formas de plantear las clases de un proyecto: la primera en base a las entidades generadas a partir del modelo Entidad-Relación de la base de datos, esta particular forma se conoce como “DBContext” [6, p. 1]; la segunda forma es generar el modelo Entidad-Relación en base a cada uno de los casos de uso y el diagrama de clases, este planteamiento se conoce como, “Code first to a new database” [7, p. 2]. En esta ocasión vamos a utilizar “Code first to a new database”, y usaremos Entity Framework que es un conjunto de tecnologías desarrolladas por Microsoft para en nuestro caso puntual, generar las entidades a partir de las clases y relacionarlas lógicamente entre si [7, p. 9]. Además de lo descrito, Entity Framework proveerá el entorno de inserción o edición de los datos a la base, es decir los servicios web se basarán en éste para realizar las operaciones con la base de datos.

Los servicios web se desarrollarán usando ASP.NET Web API 2, que provee servicios tipo REST que son basados en los estándares HTTP, usados en la gran mayoría de dispositivos electrónicos que tiene acceso a servicio de Internet. La principal ventaja de usar servicios REST y con ello sus respuestas usando el estándar JSON, es la simplicidad de su estructura, más liviana que las respuestas en XML [8, p. 6] de los servicios SOAP, vital para el ahorro de recursos, tiempos de antena y batería, tan importantes en dispositivos móviles.

View, es la ventana, muestra y recibe los datos que provienen del usuario por medio de sus controles. La principal premisa del modelo MVVM es que haya cero código dentro de cada uno de los elementos dentro del View [4, p. 33], aunque esto en la práctica es un poco complejo, la idea

que debe primar es la de hacer al formulario lo más liviano posible, evitando escribir la mayor cantidad de código posible dentro de estos.

La parte del ViewModel cumple dos funciones en el patrón MVVM:

- Ser un mecanismo de transporte, que lleve únicamente los datos necesarios al modelo y traiga de la misma forma los datos a la vista. [4, p. 33]
- Actuar como controlador sobre la vista y el modelo, recibiendo las acciones del usuario ejecutadas en la vista. [4, p. 33]

Se podría decir que la característica más explotada del patrón MVVM es que puede atarse (Binding) el View con ViewModel, mediante características propias de XAML, evitando así mediante código asignar un valor de clase a un control; XAML busca el valor y lo coloca de manera automática. [5, p. 26]

La principal ventaja de este patrón es que una vez culminado el desarrollo se puede reutilizar tanto la parte del Model como la del ViewModel para desarrollar la aplicación en otras plataformas como IOS y Android. El Model mantiene completamente todo lo relacionado a los servicios publicados, sólo las clases deben ser implementadas en la nueva solución tal como se desarrollaron. El ViewModel, al igual que las clases del Model, puede ser implementado sin cambios, usando Xamarin Platform, que permite realizar aplicación para IOS y Android usando código C# [9, p. 6]. De igual manera la parte del View, puede ser construida usando Xamarin.Forms desde Visual Studio mediante los controles que éste provee [9, p. 8].

3.3 Conocimiento requerido

Desde la planificación para el desarrollo de la aplicación, se necesitan conocimientos sólidos de Ingeniería de Software, usaremos metodologías de desarrollo ágil, en este caso Scrum para nuestro proyecto.

El correcto planteamiento y conocimiento del problema, es importante para determinar un buen alcance, esto conlleva a lograr un correcto desarrollo de clases y diagramas de casos de uso, lo que es fundamental para la óptima utilización de Entity Framework, ya que si no se realiza un diagrama de clases adecuado para la aplicación, las fallas de este se verán reflejadas en el modelo Entidad-Relación, generándose un efecto dominó en el resto del proyecto.

La estructura de los servicios web, su funcionamiento y arquitectura, son temas importantes que se deben conocer para el correcto desarrollo de la capa de servicios los cuales debe estar correctamente estructurados, y deben responder a las funcionalidades requeridas, ya que se puede caer en el riesgo de construir servicios, poco eficientes que afecten al buen desenvolvimiento del servidor y de la aplicación.

Para el desarrollo de las clases pertenecientes a la capa ViewModel, se necesita conocer mucho sobre la teoría que encierra estos elementos; cuál es su rol esperado en el funcionamiento de la aplicación, y hasta donde debe llegar la funcionalidad de las mismas, esto es importante para no caer en generar líneas de códigos extras que debiesen estar en el View o sobrecargar los ViewModel con líneas innecesarias.

Conocer la correcta forma de estructurar funciones y algoritmos, es imprescindible para el desarrollo de las clases del ViewModel, sumado a las capacidades positivas de desarrollar con C#, hacen que las clases del ViewModel sean lo suficientemente flexibles para sostener el fundamento de universalidad anteriormente planteado.

Las interfaces deben ser desarrolladas usando conocimientos de interacción hombre máquina, usando XAML con todas sus características para hacer que éstas sean intuitivas y fáciles de manejar por los usuarios.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. La tecnología es un gran aliado para resolver problemas cotidianos de maneras poco convencionales, en este caso, la mejor manera de conectar quienes tienen espacio disponible en su medio de transporte cotidiano con quien necesitan servicio de transporte, se hace a través de aplicaciones móviles y de computadores personales.
2. Si queremos conectar ofertantes con consumidores, la forma más directa es a través de medios electrónicos (“teléfonos inteligentes, tabletas o computadores personales”), que en esta época son elementos cotidianos en la vida de las personas.

Recomendaciones

1. Buscar un patrón de diseño que permita a futuro ampliar de forma sencilla la aplicación desarrollada a otras plataformas.
2. Definir correctamente los casos de uso de la aplicación, apegándolos a la realidad requerida y sobre acciones concretas, para evitar que estos se hagan muy extensos y difíciles de cumplir en tiempos normales.
3. Se debe seleccionar en base a la aplicación la forma de crear las clases del proyecto. Al principio se pensó en crearlas en base a las entidades generadas a partir del modelo Entidad-Relación de la base de datos, pero reconsiderando la naturaleza interactiva de la aplicación se desechó este primer planteamiento, y se buscó crear el modelo entidad Relación en base a las clases que se generaron analizando los diferentes casos de uso.

Futuro de la aplicación

1. Tomando en cuenta el patrón de desarrollo MVVM que promueve el concepto de universalidad para las aplicaciones, el siguiente gran paso que podemos dar es la de llevar a la aplicación de la ESPOL a otras plataformas móviles.
2. Llevar servicios como registros de materias, alquileres o pagos; a la aplicación móvil de la ESPOL.

3. Incluir respuestas a tareas o requerimientos planteados en las asignaturas registradas.

BIBLIOGRAFÍA

[1] G. d. P. Estratégica, «Rendición de Cuentas 2014,» Escuela Superior Politécnica del Litoral, Guayaquil, 2014.

[2] Uber, «Uber,» Uber, 01 Marzo 2009. [En línea]. Available: https://www.uber.com/es/. [Último acceso: 28 Septiembre 2015].

[3] EasyTaxi, «EasyTaxi,» EasyTaxi, 24 Junio 2014. [En línea]. Available: http://www.easytaxi.com/ec/. [Último acceso: 28 Septiembre 2014].

[4] P. J. J. G. Jesse Liberty, Pro Windows 8.1 Development with XAML and C#, London: Apress, 2014.

[5] I. C. Matt Baxter-Reynolds, Programming Windows Store Apps with C#, Sebastopol: O'Reilly, 2014.

[6] R. M. Julia Lerman, Programming Entity Framework DbContext, Sebastopol:
O'Reilly, 2012.

[7] J. L. &. R. Miller, Programming Entity Framework: Code First, Sebastopol: O'Reilly, 2012.

[8] B. W. Jamie Kurtz, ASP.NET Web API 2: Building a REST Service from Start to Finish, Sebastopol: Apress, 2014.

[9] C. Petzol, Create Mobile Apps with Xamarin Preview Edition 2, Redmond: Microsoft Press, 2015.