risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido/Submission: 01/06/2017
Aceitação/Acceptance: 15/09/2017

Implementación de una solución web y móvil para la gestión vehicular basada en Arquitectura de Aspectos y metodologías ágiles: Un enfoque educativo de la teoría a la práctica

Pablo Alejandro Quezada- Sarmiento [1], Santiago Mengual Andrés [2]

paquezadasa@uide.edu.ec, paquesar@alumni.uv.es, santiago.mengual@uv.es

1 Universidad Internacional del Ecuador, Docente Investigador Escuela de Informática y Multimedia, Av. Simón Bolívar y Av. Jorge Fernández, Quito, Ecuador. Universitat de ValEncia, Programa de Doctorado en Educación, Facultad de Filosofía y Ciencias de la Educación, Av. Blasco Ibáñez 30, 46010 València, España

2 Universitat de Valencia, Profesor Contratado Doctor, Facultad de Filosofía y Ciencias de la Educación, Av. Blasco Ibáñez 30, 46010 València, España

DOI: 10.17013/risti.25.98-111

Resumen: En el presente artículo se detallan el análisis, aplicación de metodologías ágiles, y tecnologías de código abierto en el desarrollo de una aplicación web y móvil, para automatizar y controlar el parque automotor de la Universidad Técnica Particular de Loja (UTPL), dicha aplicación fue desarrollada por los profesionales en formación del componente educativo Programación Web en marco del proyecto Computación Afectiva. Dentro de las funciones esenciales del aplicativo está la gestión de vehículos que implica el registro: ordenes de combustible y mantenimiento, facturas, proveedores, conductores y vehículos; además de la gestión de préstamos de vehículos que implica el registro de solicitudes, entradas, salidas y el estado de los vehículos. Como aporte adicional está la construcción del aplicativo móvil que incluye las mismas cualidades funcionales de los aplicativos webs, pero ofreciendo otra propias de un dispositivo móvil como es el uso a través de teléfonos o tabletas, facilitando la interacción y usabilidad en base al momento y lugar donde la persona lo requiera.

Palabras Clave: Aplicativo web; Educación; ICONIX; POA, Metodología

Implementation of a web and mobile solution for vehicular management based on Aspects Architecture and Agile Methodologies: An educational approach from theory to practice

Abstract: In this article, the analysis, application of agile methodologies, and technologies open source used in the development of a web and mobile application, to automated, and control the automotive fleet of the Technical University of Loja (UTPL) it was detailed; this application it was development by educational component of Web Programming in the Affective Computational Project. The purposes, the use of the aspect oriented programming paradigm (POA) and the

agile development methodology ICONIX were established. Within the essential functions of the web application is the management of vehicles that involved registration: fuel and maintenance orders, invoices, suppliers, drivers and vehicles; In addition to the management of vehicle loans that involved the registration of applications, inputs, outputs and vehicle status. As an additional contribution is the construction of the mobile application that includes the same functional qualities of web applications but offering another one of a mobile device such as use through cellphones or tablets, facilitating interaction and usability based on the time and place where people requires it. Once the web application has been implemented, users and the administrator will have the ability to access all the services that it presents in an interactive and truthful way, helping to better control, management and usability based on the time and place where the people requires it.

***Keywords**:* Education; Iconix; POA; Methodology; Web Application.

1. Introducción

Las nuevas tecnologías y paradigmas computacionales pueden ser empleados en diversos campos, entre ellos el de educación lo cual sirven para el diseño, búsqueda, presentación, intercambio y reutilización de material debido a que la tecnología permite almacenar, organizar, replicar, difundir, transformar y ser accesible, lo que conlleva al ahorro de tiempo y recursos (Area, 2005). En el área educativa se ha visto la necesidad de crear espacios donde no existan limitantes de tiempo ni capacidad; es así, que la educación usa diversas plataformas para el proceso de enseñanza - aprendizaje (Quezada, Enciso & Garbajosa, 2016). "*A nivel de la educación superior, especialmente en los campos de la Ingeniería de Software, el mundo académico muestra un entusiasmo significativo por el desarrollo de diversas competencias relacionadas a las competencias tecnológicas de ello la necesidad del adecuado uso de herramientas complementarias como son las herramientas de código libre"* especialmente enfocadas a la mejora de destrezas de programación que a su vez estén alineado a los estándares o cuerpos de conocimiento que rigen a dichas disciplinas y al contexto de la computación afectiva *(*Quezada, 2017),(Gutiérrez & Serrano ,2016) permitiendo un adecuado entorno de aprendizaje.

Una de las principales preocupaciones de la industria del software es desarrollar el talento de su recurso humano, ya que la calidad y la innovación de sus productos y servicios dependen en gran medida de los conocimientos, la capacidad y el talento de sus ingenieros en informática. El conocimiento ya existe; el objetivo es establecer un consenso sobre el subconjunto del núcleo del conocimiento que caracteriza a la disciplina de la ingeniería de software e ingenierías. De la misma manera *"La articulación de un conjunto de conocimientos es un paso esencial hacia el desarrollo de una profesión, ya que representa un amplio consenso respecto a lo que un profesional de la ingeniería de software debe conocer, en especial en las destrezas computacionales y paradigmas computacionales"* (Quezada, 2016). "*En un marco más amplio, la evolución actual de la Sociedad del Conocimiento exige un cambio de mentalidad en el profesorado para adaptar las metodologías docentes, en base a las posibilidades que ofrece el entorno digital para enseñar de forma distinta, adaptándose a las formas de aprender de los "nativos digitales*" (Poy, Mendaña & González, 2015), por ello la enseñanza de nuevos paradigmas de programación con un enfoque de metodologías agiles y basados en los principios de la computación afectiva son los pilares de la formación de los ingenieros en software.

El presente artículo aborda la temática de arquitectura del software aplicada a aspectos como paradigma en el desarrollo de aplicaciones web y móviles; enfocado a un contexto de gestión vehicular. El sistema de gestión vehicular es un sistema informático que potencializa la administración del parque automotor de la UTPL; mejorando sustancialmente la ejecución de los procesos y la obtención de información relacionada con la misma. El mejoramiento se refleja en la optimización de recursos (económicos, humanos, vehiculares, etc.); así como en una importante rapidez en el cumplimiento de los procesos. Todo esto se logró a través de Programación Orientada a Aspectos (POA) la cual es una nueva metodología de programación que aspira a soportar la separación de las propiedades para los aspectos antes mencionados. La metodología de desarrollo que se utilizó fue ICONIX, la misma que es un punto medio entre la metodología RUP y la metodología XP, todo esto enmarcado en proyecto Computación afectiva.

2. Marco teórico

"*En la actualidad las empresas que desarrollan software se han convertido en socios estratégicos de sus clientes a través de la generación de valor, mediante la entrega de productos que soportan sus diversos procesos de negocio*" (Chavarría, 2016). Si hablamos de un concepto de procesos de software hablamos de uno de los conocimientos más abstractos de la ingeniería del software, el proceso de software definido (Sommerville,2005) se ha logrado concebir como un proceso de continuo aprendizaje mediante el cual se mejora una organización a través de procesos adquiridos; así obtenemos el proceso llamado Capability Maturity Model (CMM); la última versión del CMM, ICMMI o CMM Integrated, es un instrumento de gestión de ingeniería de software formal y tradicional.

Definiciones de Aspectos. - La definición formal de "*Aspecto*" ha evolucionado desde su concepción hasta el momento. Una definición inicial, aunque todavía no se manejaba el término "aspecto", fue introducida (Joskowicz, 2008). Aplicando la definición al término actual: "*Un aspecto es una unidad que se define en términos de información parcial de otras unidades*" (Artificial, 2014). Los aspectos tienden a no ser unidades de la descomposición funcional del sistema, sino a ser propiedades que afectan la performance o la semántica de los componentes en forma sistemática"

Fundamentos de la programación orientada a aspectos. -Se puede decir que con las clases se implementan las funcionalidades principales de una aplicación (como, por ejemplo, la gestión de un almacén), mientras que con los aspectos se capturan conceptos técnicos tales como la persistencia, la gestión de errores, la sincronización o la comunicación de procesos (Barba, 2016), (Calderón, 2016). Los lenguajes orientados a aspectos definen una nueva unidad de programación de software para encapsular las funcionalidades que cruzan todo el código. Además, estos lenguajes deben soportar la separación de aspectos como la sincronización, la distribución, el manejo de errores, la optimización de memoria, la gestión de seguridad, la persistencia (Jacobson, 2004).

De todas formas, estos conceptos no son totalmente independientes, y está claro que hay una relación entre los componentes y los aspectos, y que, por lo tanto, el código de los componentes y de estas nuevas unidades de programación tiene que interactuar de alguna manera(Quintero,2000). Los puntos de enlace son una clase especial de

interfaz entre los aspectos y los módulos del lenguaje de componentes. Son los lugares del código en los que éste se puede aumentar con comportamientos adicionales. Estos comportamientos se especifican en los aspectos. El encargado de realizar este proceso de mezcla se conoce como tejedor (del término inglés weaver) (Quintero,2000).

Diseño y desarrollo de aplicaciones orientadas a aspectos. - La orientación a aspectos se centró principalmente en la implementación y codificación. pero en los últimos tiempos cada vez surgen más trabajos para llevar la separación de incumbencias a nivel de diseño (Mingueza,2010). Se propone utilizar UML (Unified Modeling Language) como lenguaje de modelado, ampliando su semántica con los mecanismos que el propio lenguaje unificado tiene para tales efectos y consiguiendo así representar la funcionalidad básica separada de los otros aspectos. Para desarrollar un sistema basado en aspectos se requiere incluir en el lenguaje base dentro de los lenguajes de aspectos y compartirse entre ambos lenguajes.

El lenguaje de los componentes debe proveer la forma de implementar la funcionalidad principal y asegurar que los programas escritos en ese lenguaje no interfieran con los aspectos. Los lenguajes de aspectos tienen que proveer los medios para implementar los aspectos deseados de una manera intuitiva, natural y concisa. El desarrollo de una aplicación basada en aspectos requiere de tres pasos:

1. Descomposición de aspectos y componentes: Descomponer los requerimientos para distinguir aquellos que son componentes de los que son aspectos
2. Implementación de las incumbencias: Implementar cada incumbencia por separado (aspectos y componentes)
3. Recomposición: Definir las reglas que permitan combinar los aspectos con los componentes (Páez,2007).

3. Metodología

"El éxito de un proyecto de desarrollo de software depende de que el producto obtenido cumpla con las especificaciones del usuario y se termine dentro del plazo y con el presupuesto establecido" (Huanca, 2017).

Diversos autores coinciden en señalar algunos requisitos que deben tener las metodologías de desarrollo:

- Visión del producto.
- Vinculación con el cliente.
- Establecer un modelo de ciclo de vida.
- Gestión de los requisitos.
- Plan de desarrollo.
- Integración del proyecto.
- Medidas de progreso del proyecto.
- Métricas para evaluar la calidad.
- Maneras de medir el riesgo.
- Como gestionar los cambios.
- Establecer una línea de meta (Salas, 2010).

Para mayor fundamento, en la tabla 1 podemos observar las características de cada una de las metodologías de desarrollo, sean ágiles o tradicionales.

Metodologias ágil és	***Metodologias tradicional és***
Se basan en heurísticas provenientes de prácticas de producción de código.	Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo
Preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente por el equipo	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso muy controlado, numerosas normas
Contrato flexible e incluso inexistente	Contrato prefijado
El cliente es parte del desarrollo	Cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10)	Grupos grandes
Pocos artefactos	Más artefactos
Menor énfasis en la arquitectura del software	La arquitectura del software es esencial

Fuente: Canós, J et al, 2005. Metodologías Ágiles.

Tabla 1 – Comparación de metodologías

En la tabla 2 se observan algunas metodologías de desarrollo y sus características las cuales han sido comparada.

El software presentado en este artículo que forma parte del proyecto Computación Afectiva, fue desarrollado por los profesionales en formación del componente educativo de Programación Web bajo los principios de la metodología Iconix (Figura 1).

Figura. 1 – Estudiantes de Ingeniería Web desarrollando prototipos bajo los principios de la Metodología Iconix.

ICONIX. - *Es una metodología pesada-ligera de Desarrollo del Software que se halla a medio camino entre un RUP (Rational Unified Process) y un XP (eXtreme Programming). Iconix deriva directamente del RUP y su fundamento es el hecho de que un 80% de los casos pueden ser resueltos tan solo con un uso del 20% del UML, con lo cual se simplifica muchísimo el proceso sin perder documentación al dejar solo aquello*

Cascada	Incremental	Protótipo	Evolutivo	Rad	Mobile D	Xp	Iconix	RUP
Breve Descripción								
Modelo que sigue una secuencia lógica y cada etapa es directamente dependiente de que se culmine la anterior	Modelo en el cual se divide previamente el proyecto en incrementos con entregas de estos en forma periódica	Consiste en entregarle a los usuarios diversos protótipos con un código cada vez más refinado del proyecto	Se enfoca en la actualización y modificación del software actual y adaptarlo a los nuevos sistemas	Método en el que se realiza una construcción rápida del prototipo	Modelo ágil de desarrollo rápido enfocado a grupos pequeños	Modelo en el que se define un plan para desarrollar y liberar software.	Proceso simplificado en comparación con otros procesos más tradicionales,	Se caracteriza por ocupar el modelo literario e incremental.
Tipos De Proyectos de Software								
Grandes empresas Proyectos gubernamentales	Juegos	Software de investigación Versiones Beta	Adaptación y mejoras de software	Aplicaciones web	Software para dispositivos móviles	Aplicaciones móviles	Está entre la complejidad del rup y la simplicidad de xp	Grandes empresas
Programador/ relación con el usuario								
Programadores experimentados. Poca relación con el usuario	Programadores experimentados Más relación con el usuario	Interactúa con el cliente para generar retroalimentación	Se relaciona de manera constante con el cliente	Trabajan pocos programadores.	Interactúa con el cliente y tiene buena relación con el grupo	Programadores con habilidades blandas y trabajo en equipo	Interactúa con el cliente, programadores	Certificados
Etapas								
Pre-análisis, Analisis, Diseño, Desarrollo Pruebas Mantenimiento	Planificación Elaboración Análisis Diseño Construcción - Entrega	Investigación preliminar Análisis y especificación diseño y construcción Evaluación modificación diseño técnico Programación y pruebas operación	Planeación Análisis de riesgo Construcción y Adaptación Evaluación del cliente	Requisitos diseño Implementación Verificación Mantención	Explotación Inicialización producción Estabilización Testeo	Definir los roles Estimar el esfuerzo Elegir que construir programar repetir	Análisis de requisito Análisis y diseño preliminar, diseño e Implementación.	Inicio Elaboración Construcción Transición
Características propias del modelo								
Se desarrolla todas sus etapas.	Cada incremento informa al siguiente y permite realizar ajustes.	Los protótipos se crena con rapidez y evolucionan a través de un proceso iterativo	Permite la realización de software	Modelo central desarrollo visual código construído extensible	Por cada función se realiza un ciclo de 3 días.	Pone énfasis en la comunicación	Su base está en los casos de uso y a partir de estos en todos los diagramas que ésta posee.	Ocupa el modelo incremental y se centra en usar casos de uso

Tabla 2 – Cuadro comparativo de las metodologías de desarrollo de mayor uso.

que es necesario. Esto implica un uso dinámico del UML de tal forma que siempre se pueden utilizar otros diagramas además de los ya estipulados si se cree conveniente (Iconix,2006).

Fases:

Análisis de Requisitos. -En esta primera fase se realiza un Modelo de Dominio, que no es más que un Diagrama de Clases simplificado. Este modelo contiene únicamente aquellos objetos de la vida real cuyo comportamiento o datos deban ser almacenados en el sistema. En la figura 2 el modelo de dominio podemos observar los respectivos atributos y el tipo de dato de los mismos, los cuales son necesarios para el correcto manejo y control del software, estos atributos fueron dados a raíz de los requerimientos otorgados por el cliente como son: préstamos vehiculares, registro de usuarios, registro de vehículos, órdenes de mantenimiento, repuesto, combustible y el registro de facturas, éstos requisitos son de vital importancia para la correcta recaudación de información, y mediante éste lograr que la aplicación web cumpla con todas las necesidades del cliente (Enciso ,2016).

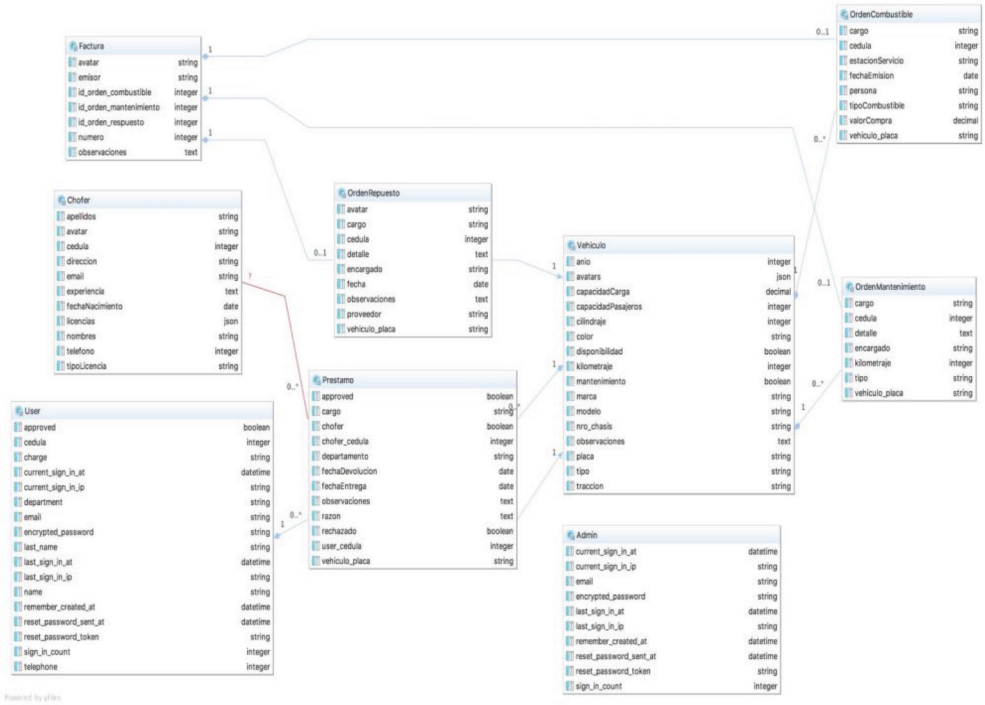
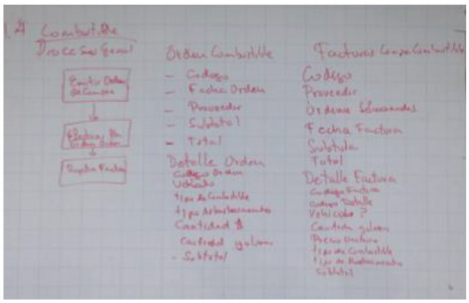
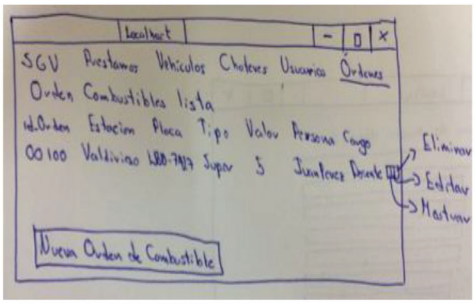


Figura 2 – Modelo de Dominio

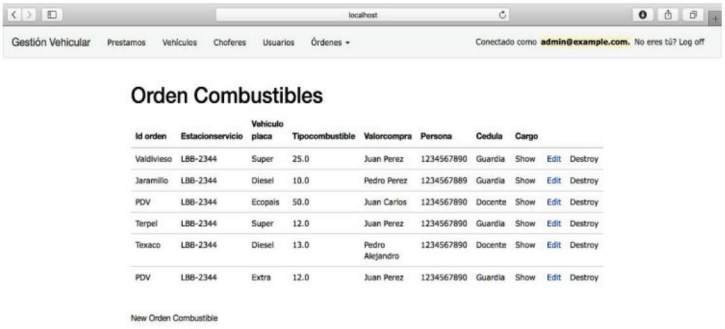
Modelo de prototipación rápida. - En la figura 3 se puede observar un bosquejo de uno de los módulos del software en este caso de orden de combustibles el mismo que sirve para aclarar las inquietudes entre el cliente y el desarrollador. Este proceso es aplicado a todos los módulos de la aplicación.



a. Análisis con cliente



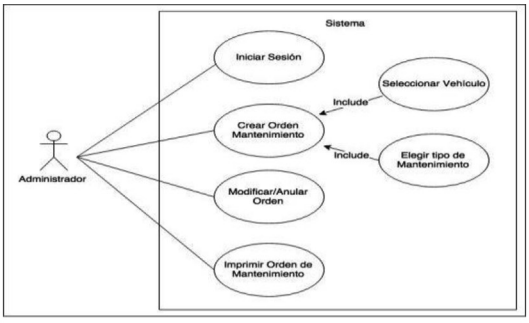
b. Esquema de Prototipo



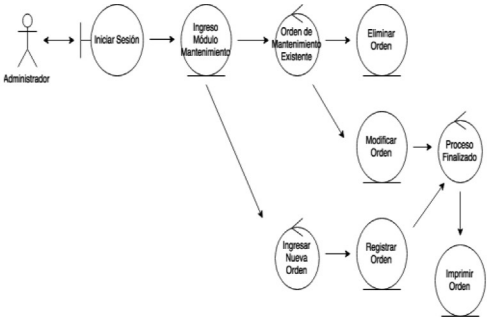
c. Pantalla inicial de Prototipo

Figura 3 – Prototipado de Modulo Orden de Combustibles de la Aplicación

Análisis y Diseño Preliminar: - A partir de cada caso de uso se obtienen sus correspondientes fichas de caso de uso que fueron aplicados a cada uno de los módulos del aplicativo. En la figura 4 se muestra un caso de uso y diagrama de robustez.



a. Caso de Uso SGV



b. Diagrama Secuencia SVG

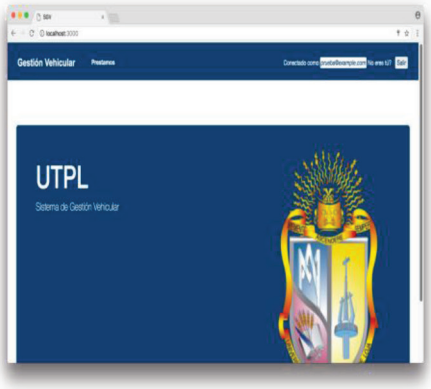
Figura 4 – Caso de uso y diagrama Sistema de Gestión Vehicular

Patrón Utilizado. - El patrón de Active Record viene implementado por defecto en el framework Ruby on Ralis, su implementación está basada en la Programación Orientada a Aspectos, lo que ha permitido un control efectivo y ordenado sobre los objetos dentro de la aplicación. El uso de Active Record ha sido extendido a lo largo de toda la aplicación, aprovechando su vasta funcionalidad en el ámbito de manipulación y almacenamiento de datos en sistemas de bases de datos relacionales, lo cual ha permitido un desarrollo conciso, simple y funcional (Comunidad Rails,2017), (Paez,2017). Los siguientes módulos de Active Record han sido implementados en la aplicación:

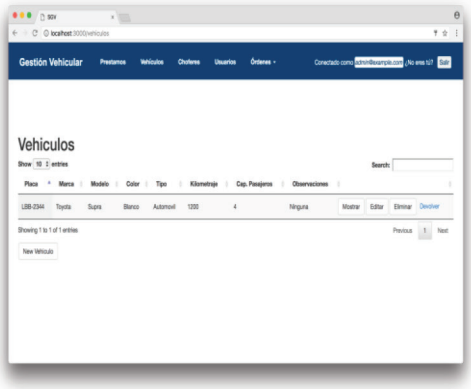
- Interfaz de consultas (Query Interface): El uso de la Interfaz de Consultas de Active Record es la principal forma de leer y manipular datos dentro de la aplicación y su base de datos, tales como registros de vehículos, préstamos, usuarios, etc. El módulo provee la generación automática de métodos sobre cada objeto para la manipulación de sus datos (crear, leer, editar y eliminar) sin el uso directo de sentencias SQL ni código de acceso a la base de datos.
- Asociaciones (Associations): Las asociaciones de Active Record han permitido crear relaciones entre modelos de Active Record (Objetos) para facilitar las operaciones entre ellos, han sido utilizadas en todos los modelos de la aplicación para crear un mapa objeto-relacional consistente y funcional, todo esto con la finalidad de establecer un manejo de datos sin redundancias ni duplicaciones y optimizando el acceso a los mismos, algunas asociaciones de Active Record que ha sido utilizadas son 'has_many'(uno a muchos), 'belongs_to'(pertecene a) y 'has_one'(uno a uno). Tal sería el caso de Vehículo tiene varios Prestamos (has_many) .
- Validaciones (Validations): Las validaciones de Active Record han sido implementadas en todos los modelos (objetos) de la aplicación para evitar el ingreso erróneo de datos y asegurar la consistencia de los datos en el sistema base de datos. Esto minimiza la tasa de errores humanos y asegura el correcto funcionamiento de la aplicación.
- Retrollamadas (Callbacks): Las retrollamadas de Active Record han sido usadas en todos los controladores de la aplicación, con la finalidad de restringir las funcionalidades y el acceso a los datos a usuarios no autorizados, en cada controlador de la aplicación se ha utilizado la retrollamada ':before_action' para exigir autenticación antes de acceder a cualquier operación. Esto ha permitido seguridad ante el acceso y operación sobre la aplicación.
- Migraciones (Migrations): Las migraciones de Active Record han sido aplicadas desde el inicio de la etapa de codificación han permitido crear un esquema de base de datos conciso y de una manera sencilla, sin usar lenguaje SQL, únicamente los métodos y comandos provistos por Active Record, se ha usado en correcciones para añadir o quitar atributos a los modelos de la aplicación, sin modificar el código de forma sustancial.

4. Resultados.

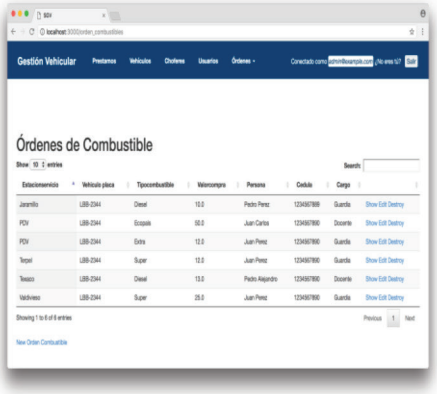
"Diseñar entornos de aprendizaje encaminados a la incorporación de pedagogías de aprendizaje activo resulta difícil" (Hood,2017); pero si se integra nuevos paradigmas, metodologías computacionales se genera óptimo resultado que permiten llevar lo teórico



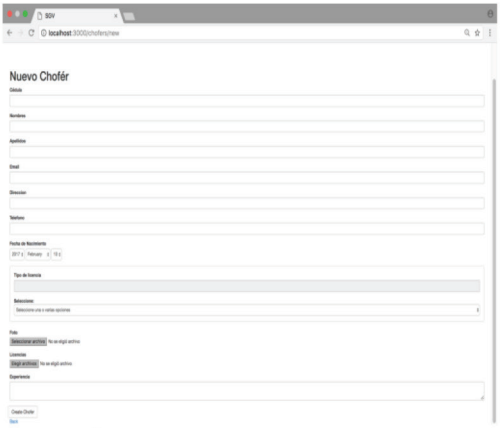
a. Intefaz inicial



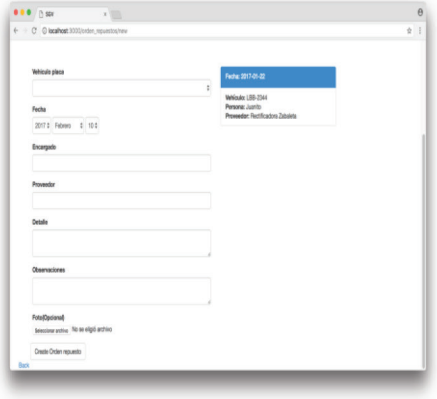
b. Módulo de Vehiculos



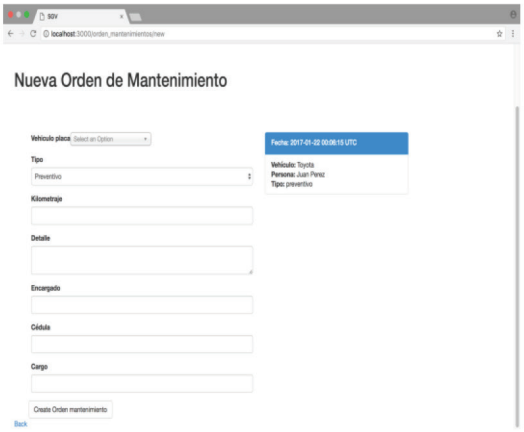
c. Módulo de Orden de Combustibles



d. Módulo de Gestión de Conductores



e. Módulo orden de repuesto



f. Módulo orden de mantenimiento

Figura 5 – Módulos Aplicación Web SGV

a la práctica. Entre los resultados más sobresaliente del Proyecto Computación Afectica: Un enfoque de lo teórico a lo práctico se destaca en la figura 5 algunos módulos del aplicativo desarrollado.

Para la creación de la aplicación móvil Android, se utilizó la herramienta AppsGeyser, que crea de una forma sencilla una WebView del sistema Android con la URL de nuestra aplicación Web, para adaptarla a dispositivos móviles. Generando un código QR con un enlace para acceder desde el dispositivo Android, una vez reconocido el QR se abre el navegador y la herramienta inicia el empaquetado(APK) del instalador para su descarga; logrando así tener la aplicación web, como aplicación móvil en cualquier dispositivo que funcione con sistema operativo Android. en la figura 6 podemos apreciar la aplicación móvil desarrollada con los principios anteriormente mencionados, todo esto en el marco de la computación afectiva.

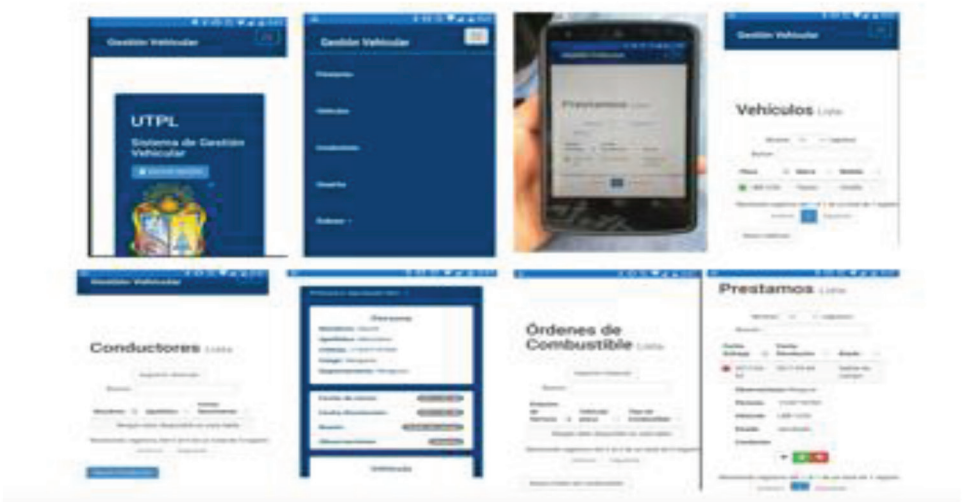


Figura 6 – Aplicación Móvil desarrollada.

5. Conclusiones

El aplicativo web y móvil durante el proceso de desarrollo bajo la metodología ICONIX y la programación orientada a aspectos, determinó un reto en base a información y manejo de todas las características que estas herramientas brindan para su correcto manejo y uso, logrando así una correcta fusión acorde a la necesidad de crear el Sistema de Gestión Vehicular.

El desarrollo de la codificación utilizando el lenguaje de programación ruby, con su framework ruby on rails, base de datos postgreSQL y el IDE RubyMine, ayudó y cumplió todas las expectativas en base a las necesidades que se planteó para el correcto desarrollo de éste software, logrando así plasmar la meta de ejercer el uso de programación orientada a aspectos, basada en ActiveRecords, herramienta ágil e intuitiva propia del framework de trabajo.

El sistema permite la administración de usuarios con su edición o eliminación; los usuarios pueden acceder a su cuenta para generar un préstamo vehicular, de la misma manera, el administrador puede, crear órdenes de mantenimiento, repuestos y de combustible, todo esto vinculado a un vehículo específico, así mismo la capacidad de registrar una factura indexándola a la orden previamente emitida.

La metodología de desarrollo ICONIX permitió un desarrollo ágil y efectivo en base a todos los requerimientos iniciales del software, ayudando en gran medida a la correcta recopilación de información, logrando así tener los requerimientos claros y precisos, los mismos que fueron base fundamental en el desarrollo del Sistema de Gestión Vehicular

El administrador tiene a su disposición el registro de todos los vehículos, órdenes, usuarios y conductores, junto a la información más relevante de los mismos que son necesarios para un correcto manejo y uso del parque automotor de la UTPL.

La validación mediante el plan de pruebas permitió demostrar la calidad, velocidad, seguridad y efectividad del software desarrollado, cumpliendo así con los requerimientos de seguridad necesarios para el uso confiable de esta aplicación.

El desarrollo del aplicativo web y movil permitió generar un trabajo colaborativo con los profesionales en formación del componente de ingeniería web, así mismo se profundizo en la enseñanza del paradigma orientado a aspectos aportando en el contexto del proyecto Computación Afectiva.

6. Trabajos Futuros

Ampliar las funcionalidades del aplicativo considerando transacciones contables y financieras; continuando con el lenguaje de programación Ruby, framework Ruby on Rails; para mantener la homogeneidad del desarrollo, la seguridad en sí del software y el proceso de enseñanza en el componente de programación web y por ende dar continuidad al proyecto computación afectiva y llevar el conocimiento de lo teórico a lo práctico.

Ampliar el contexto de analisis de la computación afectiva y como esta incide en el proceso de adquisición de una destreza computacional.

Al haber recabado información referente al paradigma de la POA (Programación Orientada a Aspectos), Computación Afectiva, Metodologías Agiles se recomienda su utilización y práctica en el contexto educativo, incidiendo en la mejora de las destrezas computacionales y el entorno de aprendizaje.

Referencias bibliográficas

Area, M. (2005). *La educación en el laberinto tecnológico. De la escritura a las máquinas digitales*. Barcelona, Octaedro EUB

Artificial, D. D. (2014). *Programación Orientada a Aspectos AOP en Spring*. Recuperado el 30 de 07 de 2016, de http://www.jtech.ua.es/j2ee/publico/spring-2012-13/apendice_AOP-apuntes.html

Barba-Guaman, L., Quezada-Sarmiento, P. A., Calderon-Cordova, C., & Lopez, J. P. O. (2017). *Detection of the characters from the license plates by cascade classifiers method*. Paper presented at the FTC 2016 - Proceedings of Future Technologies Conference, (p. 560-566). doi:10.1109/FTC.2016.7821662

Calderon-Cordova, C., Ramirez, C., Barros, V., Quezada-Sarmiento, P. A., & Barba-Guaman, L. (2017). *EMG signal patterns recognition based on feedforward artificial neural network applied to robotic prosthesis myoelectric control*. Paper presented at the FTC 2016 - Proceedings of Future Technologies Conference, (p. 868-875). doi:10.1109/FTC.2016.7821705

Canós, J. (2005). Metodologías Ágiles en el Desarrollo de Software. Valencia: Universidad Politécnica de Valencia.

Chavarría, A. E., Oré, S. B., & Pastor, C. (2016). Aseguramiento de la Calidad en el Proceso de Desarrollo de Software utilizando CMMI, TSP y PSP. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, (20), 62–77. https://dx.doi.org/10.17013/risti.20.62-77

Enciso, L., Quezada, P., Barba-Guamán, L., Solano, L., & Alarcón, P. (2016). Open drugstores mobile app doi:10.1007/978-3-319-31232-3_75

Gutiérrez Porlán, I., & Serrano Sánchez, J. (2016). Evaluation and development of digital competence in future primary school teachers at the University of Murcia. *Journal of New Approaches in Educational Research, 5*(1), 51–56. doi:http://dx.doi.org/10.7821/naer.2016.1.152

Huanca, Luis Morales, & Oré, Sussy Bayona. (2017). Factores que Afectan la Precisión de la Estimación del Esfuerzo en Proyectos de Software Usando Puntos de Caso de Uso. RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação, (21), 18–32. https://dx.doi.org/10.17013/risti.21.pi-pf

Hood Cattaneo, K. (2017). Telling Active Learning Pedagogies Apart: from theory to practice. *Journal of New Approaches in Educational Research, 6*(2), 144–152. doi:http://dx.doi.org/10.7821/naer.2017.7.237

Jacobson, I., & Ng, P. (2004). Aspect Oriented Software Development with Use Cases, (1a edición). Nueva York, USA:Addison Wesley Professional.

Joskowicz, J. (2008). *Programación Orientada a Aspectos*. Recuperado el 24 de 06 de 2016, de http://iie.fing.edu.uy/~josej/docs/Programacion%20Orientada%20Aspectos%20-%20Jose%20Joskowicz.pdf

Librosweb. (2006). Obtenido de Introducción a Ruby on rails : http://librosweb.es/libros/

Quintero, A. M. (2000). *Visión General de la Programación Orientada a Aspectos*. Recuperado el 22 de junio de 2016, de https://www.lsi.us.es/docs/informes/aopv3.pdf

Mingueza, V. (2010). *Net Development*. Recuperado el 20 de agosto de 2015, de https://victormingueza.wordpress.com/2010/06/12/introduccin-a-la-programacin-orientada-a-aspectos-aop/

Iconix Brand Group. (2016). *Manual Introductorio de Iconix*. disponible en http://ima.udg.edu/~sellares/EINF-ES2/Present1011/MetodoPesadesICONIX.pdf

Poy-Castro, R., Mendaña-Cuervo, C., & González, B. (2015). Diseño y evaluación de un juego serio para la formación de estudiantes universitarios en habilidades de trabajo en equipo. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, (spe3), 71–83. https://dx.doi.org/10.17013/risti.e3.71-83

Pons, C. G.-P. (2010). *Desarrollo de Software dirigido por modelos*. Mar del Plata: Editorial de la Universidad de la Plata.

Páez, N. M. (2007). *Utilización de programación orientada a aspectos en aplicaciones enterprise*. Recuperado el 06 de 09 de 2015, de http://web.fi.uba.ar/~npaez/content/tesis_npaez.pdf

Quezada, P., Garbajosa, J., & Enciso, L. (2016). *Use of standard and model based on BOK to evaluate professional and occupational profiles*. doi:10.1007/978-3-319-31232-3_27

Quezada-Sarmiento, P., Enciso-Quispe, L., Garbajosa, J., & Washizaki, H. (2016). Curricular design based in bodies of knowledge: Engineering education for the innovation and the industry. Paper presented at the *Proceedings of 2016 SAI Computing Conference, SAI 2016*, (pp. 843-849). doi:10.1109/SAI.2016.7556077

Quezada-Sarmiento, P. A., Mengual-Andrés, S., Enciso-Quispe, L., & Espinoza, V. (2017). Used and interaction in technological platforms of open source to improve the linguistic competence in computer engineers. In *Proceedings of Iberian Conference on Information Systems and Technologies, CISTI*, doi:10.23919/CISTI.2017.7975748

Quezada P., Enciso.L. & Garbajosa, J. (2016). Use of body knowledge and cloud computing tools to develop software projects based in innovation. In *Proceedings of 2016 IEEE Global Engineering Education Conference (EDUCON),* (pp. 267-272). Abu Dhabi. doi: 10.1109/EDUCON.2016.747456 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7474564&isnumber=7474513

Quezada-Sarmiento, P. A., Chango-Cañaveral, P. M., López-Criollo, J., Pacheco-Viteri, F. A., & Enciso, L. (2017). Design of a wireless network of temperature and lighting sensor for gastronomic laboratories under the principles of agile scrum methodology. *Espacios*, 38(46).

Sommerville I., (2005), *Software Engineering, (seventh ed)*. Pearson Addison Wesley.

Salas, T. -R. (2010). *Revistas de Investigación UNMSM*. Obtenido de http://revistasinvestigacion.unmsm.edu.pe/index.php/idata/article/view/6191/5386

Vidal, C. L., Hernández, D. D., Pereira, C. A., & Del Río, M. C. (2012). Aplicación de la Modelación Orientada a Aspectos. *Información tecnológica*, 23(1), 3–12. https://dx.doi.org/10.4067/S0718-07642012000100002