Sistema de información bajo la metodología ágil OpenUP para la gestión y evaluación de los grupos organizados de la Iglesia Adventista Universitaria Villa Unión Tarapoto

Quiroz Menor Persy[1], Tocto Cano Esteban[2]
Recibido 2 de septiembre de 2016, Aceptado 4 de octubre de 2016
Received: September 2, 2016 *Accepted: October 4, 2016*

Resumen

La presente investigación tuvo el objetivo de implementar un sistema de información web bajo la metodología de desarrollo ágil OpenUP y la NTP/IEC 12207 para la gestión y evaluación de los grupos organizados de la Iglesia Universitaria Villa Unión de la Universidad Peruana Unión, Filial Tarapoto. El sistema de información se desarrolló integrando el proceso de desarrollo de la NTP ISO/IEC 12207 en las fases de concepción, elaboración, construcción y transición de la metodología de desarrollo ágil OpenUP del Eclipse Process Framework, con el objeto de obtener un software que responda a estándares de calidad y cumpla con los requerimientos de los usuarios. Se aplicó el patrón de diseño de la arquitectura Modelo Vista Controlador (MVC) con la Programación Orientada a Objetos (POO) y conexión a base de datos PostgreSQL versión 9.0 mediante el patrón Singleton que permite dar una mejor respuesta a las peticiones de los usuarios. El sistema de información fue desarrollado en el lenguaje de programación Java, uso de JavaScript, estilos CSS3 y HTML5. El sistema posee las características funcionales para una buena gestión y evaluación de los grupos organizados. Los líderes de la iglesia y grupos gestionan la información por semestre académico, permitiendo tomar decisiones en base a medidores de desempeño (KPI) en tiempo real que permiten medir el desempeño de cada grupo organizado. En conclusión, esta herramienta tecnológica permite gestionar eficazmente a los grupos organizados.

Palabras clave: Metodologías ágiles, OpenUP, PostgreSQL, Singleton, Modelo Vista Controlador.

[1] Ingeniero de Sistemas. Universidad Peruana Unión. email: persy.quiroz@upeu.edu.pe
[2] Ingeniero de Sistemas. Magíster en Sistemas. Universidad Peruana Unión. email: estocan@upeu.edu.pe

Abstract

This research had the objective of implementing a web information system under the agile OpenUp methodology and the NTP/IEC 12207 for the management and evaluation of the organized groups of the Adventist university church Villa Unión of the Peruvian Union University subsidiary Tarapoto. The information system was developed integrating the process of development of the NTP ISO / IEC 12207 in the phases of conception, elaboration, construction and transition of the methodology of development of agile OpenUP of the Eclipse Process Framework, in order to obtain software that respond to the quality standards and fulfill the user's requirements. The design pattern of the architecture Model View Controller (MVC) was applied with Object Oriented Programming (OOP) and connection to the database PostgreSQL 9.0, using the Singleton pattern that allows a better response to the requests of the users. The information system was developed in the Java programming language, use of JavaScript, CSS3 and HTML5 styles. The system has the functional characteristics for a good management and evaluation of the organized groups. Church and groups leaders manage the information by academic semester, allowing decisions to be made based on performance measures (KPI) in real-time that allows to measure the performance of each organized group. In conclusion, this technological tool allows effective management of organized groups.

Keywords: Agile methodologies, OpenUp, PostgreSQL, Singleton, Model View Controller.

INTRODUCCIÓN

La tecnología forma parte importante de la vida del ser humano, la información es uno de los activos más importantes que poseen las organizaciones. A pesar de que la tecnología está presente, muchas de las empresas no cuentan con los mecanismos eficaces que le permitan optimizar sus diversos procesos, a pesar que poseen los recursos donde pueden operar las aplicaciones, las razones son muchas, una de ellas es una falta de cultura de innovación, se conforman con lo que tienen y no prueban nuevas soluciones que pueden ser más eficaces, mejorando su rendimiento y eficacia y cubriendo más allá de las expectativas de los clientes finales. Los sistemas de información son claves para el fortalecimiento de los procesos de cualquier rama a la que se dedica la organización.

La Iglesia Universitaria está organizada por escuelas sabáticas, y las mismas en grupos pequeños, además la iglesia cuenta con otros tipos de grupos. Los grupos pequeños son conjuntos de personas en promedio de 8 a 20 personas que realizan actividades diversas, con objetivos bien marcados. Entre los cuales figuran la conservación de los integrantes o feligreses, el aprendizaje continuo de las Sagradas Escrituras, la realización de actividades misioneras para el cumplimiento de la misión adventista. Todos estos objetivos giran en tres ejes principales: comunión,

relacionamiento y misión. Cada grupo cuenta con un líder, colíder y consejero y demás cargos, poseen una lista de integrantes, en su mayoría estudiantes universitarios. Todos los procesos se controlan de forma manual y no automatizada. Entre los problemas existentes se encuentran los siguientes: no se cuenta con una ficha histórica por cada persona donde esté especificado su asistencia a las reuniones de los grupos, su participación y evaluación respectiva, lo cual genera pérdidas económicas a la iglesia e institución por becas de feligresía asignada a personas que dicen ser adventistas pero no participan de las reuniones y actividades de la iglesia, no se cuenta con un registro automatizado de asistencia por lo que no se sabe qué personas asistieron y no asistieron a cada reunión, ya que esto permitiría las visitas de capellanía a cada alumno con problemas espirituales, cada grupo no cuenta con un registro del programa para cada evento lo cual genera imprevistos en el evento, no se cuenta con una tabla comparativa automatizada, lo cual genera un proceso lento de cálculo al departamento de Escuela Sabática de la iglesia y líder del grupo, no se tiene control a nivel general de los estudios bíblicos que realizan. Además se desconoce los siguientes indicadores de gestión que ayude a la toma de decisiones tales como: personas con más asistencias puntuales, tardanzas, faltas injustificadas, faltas justificadas, integrantes dando estudios bíblicos, % de personas en grupos pequeños por entidades académicas (facultades, escuelas profesionales), personas más participativas, % de estados de asistencia por entidades académicas, promedio de evaluación por entidades académicas, personas con mejor evaluación, análisis de asistencias por facultad, análisis situacional por persona de manera que le permite decidir integrarse y comprometerse con la misión adventista, termómetro de Escuela Sabática por grupo, entre otros reportes importantes.

MARCO TEÓRICO

Metodologías ágiles

Son formas o técnicas de desarrollar software de buena calidad en tiempos optimizados, poseen fases, principios, disciplinas, roles y prácticas que permiten una buena gestión de los proyectos de software. Según Rao, Hiranmayi, Priya, & Ravi (2014), una "metodología ágil es una metodología de desarrollo de software usado para mejorar la velocidad en la entrega de proyectos de software" y define a la metodología ágil como "un enfoque iterativo e incremental (evolutivo) para el desarrollo de software que se realiza de una manera altamente colaborativa por equipos autoorganizados dentro de un marco de gobierno efectivo con el objeto de producir software de alta calidad a un costo efectivo y adecuado de manera que responda a las necesidades cambiantes de las partes interesadas". En los últimos años la mayor parte de las organizaciones como Google, Microsoft y Siemens han adoptado la metodología ágil en algunos proyectos importantes. Además dicen Khan, Rehman, Khan, Shah, & Khan (2016): "las metodologías ágiles han conseguido una gran reputación en los últimos años para la construcción de productos de software", siendo el objetivo principal entregar frecuentemente software de alta calidad y de

acuerdo a las necesidades de los clientes, permite a los equipos de desarrollo hacer entregas dentro del presupuesto y el tiempo programado. Las técnicas de desarrollo ágil producen mejores resultados que las técnicas tradicionales. Las metodologías ágiles se basan en producir resultados de manera incremental y en enfoques iterativos. Para gestión de requisitos aplican técnicas innovadores tales como entrevistas grupales. Existe documentación limitada, mucha participación de los clientes (forman parte activa dentro del equipo de desarrollo) y se da mucha importancia a las personas en vez de los procesos. En la Tabla 1 se muestra las principales diferentes entre metodologías ágiles y las tradicionales.

Tabla 1
Comparación de los enfoques tradicionales con ágil.

Tipo	Enfoque tradicional	Enfoque Ágil
Desarrollo del ciclo de vida	Secuencial o lineal	Iterativo e incrementales
Estilo de desarrollo	Anticipado	Adaptado
Requisitos	Bien informado tempranamente, claramente definido y documentado	Emergente, cambio rápido y descubierto durante el proyecto
Arquitectura	Arquitectura en promedio pesada para las necesidades actuales y futuras	Usa el principio YAGNI (no agregar funcionalidad que no se va a necesitar
Administración	Centrada en el proceso, comando y control	Las personas motivadas, un buen liderazgo y la eficiente colaboración
Organización del equipo	Equipos preestructurados	Equipos autoorganizados

Fuente: (Rao et al., 2014).

Para el éxito de proyectos de software existen varios factores, uno de ellos es la experiencia de los desarrolladores con el uso de las tecnologías que se aplican, un buen análisis de requerimientos que resultan en el diseño eficiente de las bases de datos y el plan de ciclos de vida de los proyectos, por medio del uso de metodologías ágiles.

Según Dhir & Sarraf (2016) entre las metodologías ágiles podemos encontrar: feature driven development (desarrollo impulsado por funciones), dynamic software development methods (métodos de desarrollo de software dinámico), lean software development (desarrollo ágil de software), crystal methodologies. Y Kamthan (2013) menciona a XP (Programación Extrema), OpenUP (Proceso Abierto Unificado abierto), y Scrum.

Open Unified Process (OpenUP)

Según Loniewski (2011)to reduce time and cost of development, augmenting flexibility and interoperability. However, many techniques and approaches that have been introduced are of little use when not provided under a formalized and well-documented methodological umbrella. A methodology gives the process a well-defi-

ned structure that helps in fast and efficient analysis and design, trouble-free implementation, and finally results in the software product improved quality. While MDD and SOA are gaining their momentum toward the adoption in the software industry, there is one critical issue yet to be addressed before its power is fully realized. It is beyond dispute that requirements engineering (RE argumenta que "OpenUP es un proceso de desarrollo mínimamente suficiente, pero completa, ágil y extensible, fue desarrollado por los empleados de IBM Rational". Posee fases, disciplinas, documentos, productos de trabajo, actividades y flujos de trabajo. OpenUP se une a muchos principios que ayudan a los equipos a ser más eficaz en el desarrollo de software, tales como: la filosofía ágil que se centra en la naturaleza colaborativa de desarrollo de software, la extensibilidad que permite que el proceso se extienda o adaptarse a las necesidades específicas de un proyecto u organización. El proceso de la metodología OpenUP es dividido en cuatro fases: Concepción, Elaboración, Construcción y Transición. El proyecto es dividido en varias iteraciones, para cada iteración hay un plan semanalmente, dicho plan o iteración es divido en ítems o tareas diarias que llegarán a ser micro-incrementos en producto. En cada iteración resulta un demo o producto funcional. Los roles que posee la metodología son: analista, arquitecto, desarrollador, jefe de proyecto, stakeholders y tester o probador. Y las disciplinas que propone son: requerimientos, arquitectura, desarrollo, pruebas, administración de configuración y cambio, y administración del proyecto.

Fases de OpenUP

Concepción: el objetivo de esta fase es comprender el alcance y los objetivos del proyecto, se evalúa si es o no viable. Hay cuatro objetivos para evaluar el alcance, los objetivos y la viabilidad siendo los siguientes: entender qué construir, identificar la funcionalidad clave del sistema, determinar al menos una posible solución y entender el costo, el cronograma y los riesgos asociados al proyecto.

Elaboración: el objetivo de esta fase es establecer la línea base de la arquitectura del sistema y proporcionar una base estable para el gran esfuerzo de desarrollo de la siguiente fase. Hay objetivos para esta fase que le ayudan a direccionar los riesgos asociados con los requisitos, la arquitectura, los costos y el cronograma que son los siguientes: obtener un entendimiento más detallado de los requisitos, diseñar, implementar, validar y establecer la línea base para la arquitectura y mitigar los riesgos esenciales y producir un cronograma exacto y unos costos estimados.

Construcción: se enfoca en el diseño, la implementación y la prueba de las funcionalidades para desarrollar un sistema completo, el objetivo es desarrollar la aplicación basada en la arquitectura establecida. Los objetivos para la fase de Construcción que nos ayudan a tener un desarrollo con costo eficiente de un producto completo, una versión operativa del sistema que pueda ser entregada a la comunidad de usuarios son: desarrollar iterativamente un producto completo que esté listo para hacer transición a la comunidad de usuarios y minimizar el costo de desarrollo, y alcance algún grado de paralelismo.

Transición: permite asegurarse que el software está listo para entregar a los usuarios finales. Los objetivos para la fase de Transición que ayudan a afinar elegantemente la funcionalidad, el desempeño y la calidad total de la versión beta del producto desde el final de la fase previa son: La prueba beta valida que las expectativas del usuario sean satisfechas, lograr que los stakeholders concuerden en que la implementación ha terminado y mejorar el desempeño en futuros proyectos a través de lecciones aprendidas.

Entre los principales entregables en OpenUP, se detallan en la siguiente Tabla 2:

Tabla 2
Principales documentos que se desarrollan en OpenUP.

Fase	Actividad	Documento
Inicio (concepción)	• Iniciación del proyecto • Planeación del proyecto • Identificación de requerimientos	• Visión del sistema • Plan de desarrollo de software • Especificación de requisitos de software
Elaboración	• Desarrollo de la arquitectura	• Documento de arquitectura del sistema
Construcción	• Definición de pruebas de la solución • Construcción de la solución (Traducción de pruebas a código y desarrollo del sistema).	• Plan de pruebas de software • Creación de archivos con los escenarios de prueba • Codificación de la solución basándose en el comportamiento definido en las pruebas
Transición	• Despliegue de la solución	• Manual de instalación • Manual de usuario

Fuente: (Ríos, Hinojosa, & Delgado, 2013).

Comparación de metodologías ágiles: se tomaron las metodologías ágiles OpenUP, XP, SCRUM y Enfoque a reglas de negocio para hacer comparaciones entre sus fases, roles prácticas y disciplinas, la Tabla 3 muestra en detalle la comparación.

Tabla 3
Comparación entre metodologías ágiles.

	OpenUP	XP	SCRUM	Enfoque de Reglas de Negocio
Procesos (Fases)	• Concepción • Elaboración • Construcción • Transición	• Exploración • Planificación • Iteraciones • Producción • Mantenimiento • Cierre del proyecto	• Presprint • Sprint • Postsprint	• Definir el alcance del negocio • Descubrimiento • Análisis • Diseño • Implementación y prueba
Roles	• Analista • Arquitecto • Desarrollador • Jefe de proyecto • Stakeholders • Tester o probador	• Cliente • Programador • Probador • Rastreador • Coach o tutor • Consultor • Gestor o Manager	• Product Owner • Scrum Master • Equipo • Usuarios • Stake-holders o interesados Managers	
Prácticas / disciplinas	• Requeri-mientos • Arquitectura • Desarrollo • Prueba • Adminis-tración de configu-ración y cambio • Adminis-tración del Proyecto	• Planeamiento del Juego • Historias de usuario • Pruebas • Programación en parejas • Refactorización • Diseño simple • Propiedad colecti-va del código • Integración con-tinua • Cliente en el lugar • Entregas pequeñas • 40 horas a la semana • Estándares de código • Metáfora	• Pila de Producto • Planifica-ción de Sprint • Reuniones • Revisiones • Retrospec-tiva	• Programación declarativa • Separación de los datos y la lógica. • Conocimiento centralizado • Simplicidad • Facilita el desarrollo de la • Aplicación • Agilidad

Valores / principios	• Colaborar para alinear los intereses y compartir el conocimiento • Centrarse inicialmente en la arquitectura para minimizar los riesgos y organizar el desarrollo. • Balance de conflicto de prioridades para maximizar los valores de los interesados • Involucrar a las partes interesadas para la continua retroalimentación y desarrollo	• Comunicación • Retroalimentación • Simplicidad • Coraje	• Flexibilidad a cambios • Reducción del Time to Market • Mayor calidad del software • Mayor productividad • Maximiza el retorno de la inversión (ROI) • Predicciones de tiempos • Reducción de riesgos

Fuente: elaboración propia.

Norma Técnica Peruana ISO/IEC 12207

Fue elaborada por el Comité Técnico de Normalización en Ingeniería de Software y Sistemas de Información en el año 2003, esta norma utilizó como antecedente la Norma Internacional ISO/IEC 12207:1995/Amd 1:2002: Information technology Software life cycle processes La presente Norma Técnica Peruana presenta cambios editoriales referidos principalmente a terminología empleada propia del idioma español y ha sido estructurada de acuerdo a las Guías Peruanas GP 001:1995 y GP 002:1995, fue aprobada por Resolución Nº 0048-2004/CRT-INDECOPI.

El objetivo de la norma es establecer un marco de referencia común para los procesos del ciclo de vida del software, con una terminología bien definida a la que puede hacer referencia a los desarrolladores de software. Contiene procesos, actividades y tareas para aplicar durante la adquisición de un sistema que contiene software, un producto software puro o un servicio software y, durante el suministro, desarrollo, operación y mantenimiento de productos software.

La NTP ISO/IEC 12207, abarca los procesos del ciclo vida del software tales como:

- Procesos principales: adquisición, suministro, desarrollo, operación y mantenimiento.
- Procesos de apoyo: documentación, gestión de la configuración, aseguramiento de la calidad, verificación, validación, revisión conjunta, auditoría y solución de problemas.
- Procesos organizativos: gestión, infraestructura, mejora y recursos humanos.

Integración de la metodología ágil OpenUP y el proceso de desarrollo de la NTP ISO/IEC 12207: el proceso de desarrollo son las actividades que se realizan para la implementación de software, al igual que la metodología ágil está orientada al mismo proceso. En la que se muestra la integración del proceso de desarrollo con la metodología ágil.

Tabla 4
Integración del proceso de desarrollo de la NTP ISO/IEC 12207 con OpenUP.

NTP/IEC 12207 (Proceso de desarrollo)	OpenUP
• Análisis de los requerimientos del sistema.	• Concepción • Entender qué construir • Identificar la funcionalidad clave del sistema • Determinar, al menos, una posible solución • Entender el costo, el cronograma y los riesgos asociados al proyecto
• Diseño de la arquitectura del sistema • Análisis de los requerimientos del software • Diseño de la arquitectura del software • Diseño detallado del software	• Elaboración • Obtener un entendimiento más detallado de los requisitos • Diseñar, implementar, validar y establecer la línea base para la arquitectura • Mitigar los riesgos esenciales y producir un cronograma exacto y unos costos estimados
• Codificación y pruebas del software • Integración del software • Pruebas de calificación del software • Integración del sistema • Pruebas de calificación del sistema	• Construcción • Desarrollar iterativamente un producto completo que esté listo para hacer transición a la comunidad de usuarios • Minimizar el costo de desarrollo y alcance algún grado de paralelismo

	• Transición
	• La prueba beta valida que las expectativas del usuario sean satisfechas
	• Lograr que los Stakeholders concuerden en que la implementación ha terminado
• Instalación del software	• Mejorar el desempeño en futuros proyectos a través de lecciones aprendidas
• Apoyo de aceptación del software	

Fuente: elaboración propia.

Patrones de diseño: según Acosta (2013) dice que "los patrones de diseño se pueden definir como esquemas predefinidos o conjunto de estrategias aplicables en diversas situaciones, de forma que el analista se asegura de que el diseño que está aplicando tiene ciertas cualidades que le proporcionan calidad". Según Ting, Hsueh, Yang, & Chang (2015) argumentan que los patrones de diseño han traído muchas influencias sobre la calidad de software, los patrones de diseño se acumulan de la experiencia de muchos desarrolladores cuando se produce el mismo problema en el diseño de software, asegurando además una arquitectura flexible.

Según Issaoui, Bouassida, & Ben-abdallah (2016) existen los patrones de diseño: Abstract Factory, Adapter, Bridge, Builder, Chain of Responsibility, Command, Composite, Decorator, Facade, Factory Method, Flyweight, Interpreter, Iterator, Mediator, Memento, Observer, Prototype, Proxy, Singleton, State, Strategy, Template Method, Visitor.

> **Singleton:** patrón de diseño que permite instanciar una clase una sola vez, siendo un solo punto de acceso hacia ella. En java se usa el atributo static que permite hacer una clase estática, guardando la instancia en memoria del servidor. En esta investigación se aplicó este patrón para la conexión de base de datos, el primer usuario abre la conexión por medio de la autenticación del sistema a la base de datos, y todas las peticiones de allí en adelante se hacen por medio de esta instancia de manera sincronizada, resulta muy efectivo ya que no es necesario estar validando la autenticación a la base de datos en cada momento de la petición, los tiempos de respuestas son más eficientes en recursos y tiempo.

Patrones de arquitectura de software: *Suárez & Gutiérrez (2016),* basándose en el estándar IEEE 1471-2000, definen a un patrón de arquitectura como "la organización fundamental de un sistema incorporando sus componentes, las relaciones entre ellos y el ambiente, y los principios que orientan su diseño y evolución que busca solucionar problemas de adaptabilidad de requerimientos, rendimiento, modularidad y acoplamiento de los componentes de una aplicación".

Según Xu & Liang (2014) existen varios patrones entre los cuales figuran: Blackboard, Pipe-and-Filter, Reflection. Entre los objetivos de los patrones de diseño es la reducción de la complejidad del diseño de software y desarrollar el software de manera ordenada.

MVC (Modelo Vista Controlador): patrón que separa la lógica de negocio de la interfaz de usuario, permitiendo la reutilización de código y flexibilidad. Donde el modelo es el encargado de los datos, consultando la base de datos, permitiendo las actualizaciones, consultas, búsquedas, etc. El controlador recibe las peticiones de los usuarios y se encarga de solicitar los datos al modelo y de comunicárselos a la vista. La vista es la representación visual de los datos, todo lo que tenga que ver con la interfaz gráfica. Ni el modelo ni el controlador se preocupan de cómo se verán los datos, esa responsabilidad es únicamente de la vista.

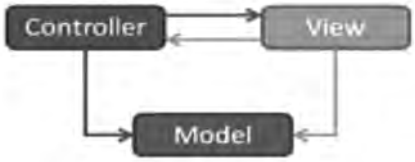

Figura 1. Patrón de arquitectura MVC - (Qureshi & Sabir, 2013).

MATERIALES Y MÉTODOS

El diseño de la investigación está dada por la siguiente metodología aplicada, como se muestra la Figura 2.

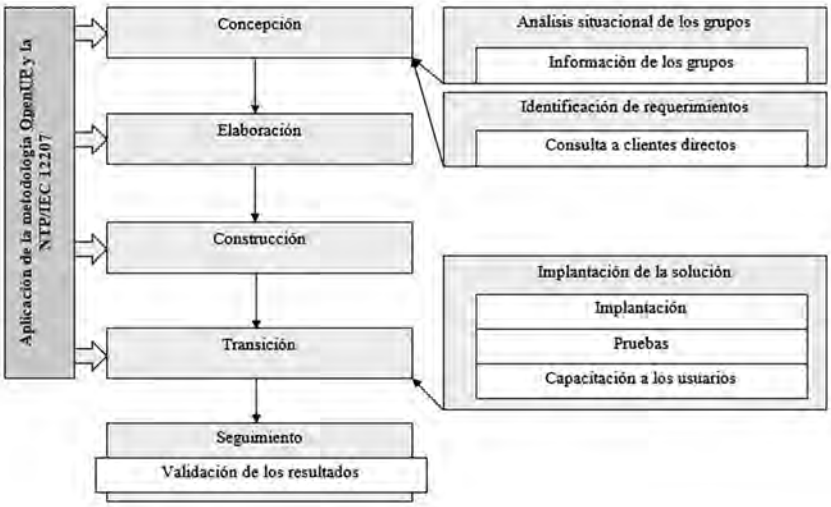

Figura 2. Metodología de investigación – Fuente: elaboración propia.

Tipo de investigación

Aplicada: esta investigación está encaminada a la solución de problemas prácticos, definidos y específicos. También porque se aplica el conocimiento para resolver problemas de cuya situación depende el beneficio de individuos (pastores, capellanes, líderes) y la comunidad (integrantes de los grupos e iglesia en general), mediante el uso de herramientas tecnológicas.

Descriptiva: porque se analiza, recolecta y valida los datos (información) de los procesos relacionados con los grupos organizados, con el objetivo de la obtención de requerimientos que llegarán a ser parte de las características funcionales de la solución propuesta.

Construcción de la propuesta

Concepción

Información de los grupos organizados (análisis situacional de los grupos): se investigó la dinámica de los grupos organizados, la información relacionada con la iglesia y las facultades de la universidad, se conoció el motivo de la existencia del grupo organizado, su misión y su visión como grupo y como institución.

Diagrama de actores: en la Figura 3 se muestran a los actores que interactúan en los procesos de los grupos organizados. Dichos actores son los usuarios potenciales del sistema.

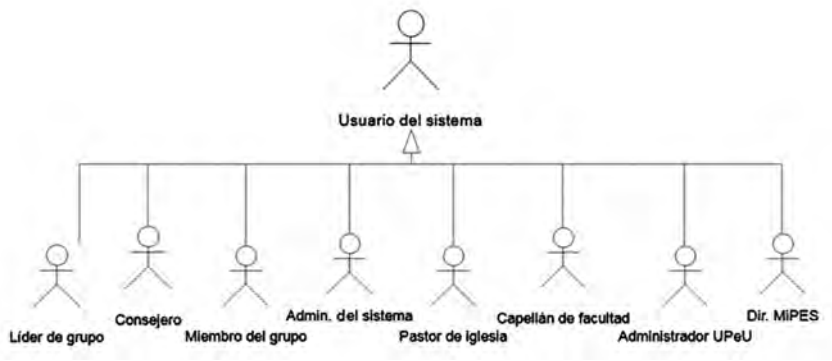

Figura 3. Actores del sistema

Diagrama de casos de uso de requerimientos: muestra los usuarios que interactúan con el sistema y las actividades que realizan cada uno de ellos en el sistema, un caso de uso es un procedimiento o actividad que hacen los usuarios.

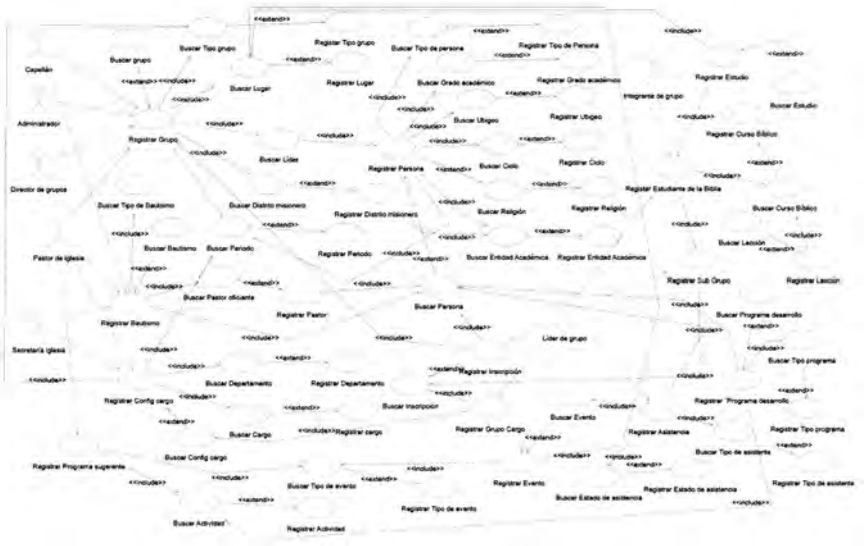

Figura 4. Diagrama de casos de uso de requerimientos del sistema.

Diagrama de paquetes del sistema: el sistema consta de siete (07) paquetes principales que vienen a ser los módulos del sistema.

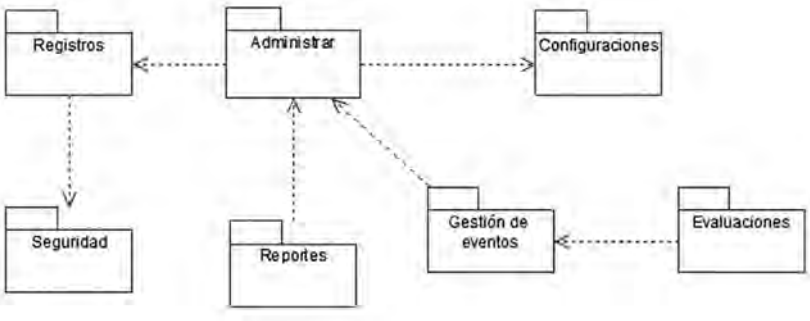

Figura 5. Diagrama de paquetes del sistema.

Elaboración

Arquitectura del sistema: la arquitectura del sistema está basada sobre el patrón diseño de Modelo Vista Controlador (MVC). Se consideró la parte del modelo en archivos .java, donde definen los métodos accesores de envío y recepción, dicho modelo representa o simula el diseño de base de datos, el controlador es que procesa las peticiones enviadas de la vista y hacia la base de datos y, al mismo tiempo, hace la devolución de la petición trabajando juntamente con el modelo. La vista está en archivos .jsp, que permiten visualizar vía HTML las respuestas del servidor, ya sea de formularios de envío o reportes. El patrón Singleton trabaja en la conexión de base de datos, con el método de abrir la conexión y manteniéndola para todos los usuarios que se conectan al sistema, esto permite un menor tiempo de respuesta a las peticiones desde la vista, haciéndole más eficiente.

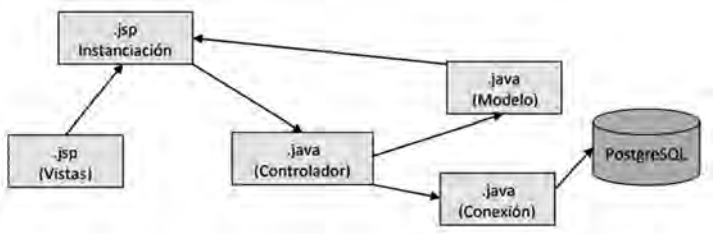

Figura 6. Arquitectura del sistema -Modelo Vista Controlador (MVC).

Diagrama de base de datos: contiene todas las tablas de base datos que están relacionadas entre sí, permitiendo guardar la información de manera organizada. El sistema de información envía y recibe los datos hacia la base de datos respectiva.

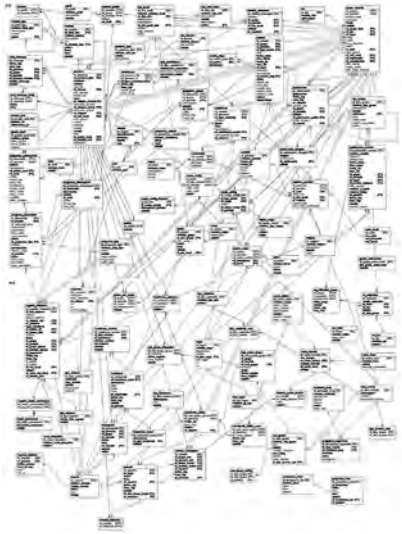

Figura 7. Diagrama de base de datos - Sistema de Grupos Organizados.

Construcción

Bootstrap: framework, a nivel de interfaz de usuario, permite la adaptación de las vista del sistema a diferentes tamaños de pantalla, permitiendo acceder al sistema que va desde un equipo completo de mesa, hasta un dispositivo móvil. Bootstrap está constituido por librerías JavaScript y archivos CSS.

Figura 8. Página de inicio, vista en pantalla grande y pequeña con la tecnología Bootstrap.

Iteraciones: Las iteraciones vienen a ser las metas de desarrollo del sistema, donde de cada iteración se obtiene una versión funcional del sistema. A continuación se detallan las iteraciones desarrolladas.

Iteración 1: la primera fase se enfocó en la implementación del proceso de autenticación y el registro de los grupos organizados, que viene a ser la base del panel de administración de cada grupo. La lista de casos de uso desarrollados son: Iniciar Sesión, Cerrar Sesión, Cambiar Contraseña, Crear Usuario Personal, Restaurar Contraseña, Crear Grupo, Modificar Grupo, Eliminar Grupo, Crear Tipo de grupo, Modificar Tipo de grupo, Crear Lugar de reunión, Modificar Lugar de reunión, Asignar Lugar de reunión al grupo, Asignar Grupo Padre, Crear Distrito Misionero, Modificar Distrito Misionero, Asignar Distrito Misionero, Ver Lista de grupos, Crear Periodo, Modificar Período, Eliminar Período y Ver Tipo de grupo.

Iteración 2: abarca el registro de personas y la inscripción de la persona al grupo organizado. Todo esto permite tener la data de cada integrante del grupo y la organización de cada grupo. Abarca los siguientes casos de uso: Crear Persona, Modificar Persona, Buscar Ubigeo, Buscar Entidad Académica, Buscar Ciclo, Buscar Período, Buscar Religión, Buscar Grado Académico, Buscar Tipo de persona, Crear Líder, Modificar Líder, Asignar Líder, Crear Integrante, Buscar Integrante por Nombres, código o Documento de Identidad, Añadir Integrante al Grupo, Modificar Integrante, Eliminar Integrante, Ver fotos de integrantes y Ver lista de integrantes.

Iteración 3: abarca la configuración de cargos, la actualización de datos del grupo, el registro de cargos en el grupo, la configuración de programas sugeridos y el registro de bautismos. Abarca los siguientes casos de uso: Crear Tipo de evento, Modificar Tipo de evento, Ver Tipo de grupo, Configurar Cargos por Tipo de grupo, Ordenar Cargos, Crear Actividad, Modificar Actividad, Eliminar Actividad, Asignar Actividad a Tipo de evento, Crear Tipo de asistente, Eliminar Tipo de asistente, Buscar Tipo de asistente, Buscar Responsable de cargo, Eliminar Responsable de cargo, Ver Cargo, Crear Bautismo, Modificar Bautismo y Buscar Pastor Oficiante.

Iteración 4: abarca la creación de eventos, elaboración del programa, control de asistencia y la evaluación a los asistentes a los eventos. Abarca los siguientes casos de uso: Ver Evento, Buscar Evento por Período y Mes, Crear Evento, Buscar Tipo de evento por Tipo de grupo, Ver programa de evento, Crear Actividad en Programa Desarrollo, Buscar Actividad por Nombre, Buscar Tipo de programa, Buscar Actividades por Tipo de evento, Modificar Actividad, Eliminar Actividad, Buscar Responsable en Persona, en Integrantes y Grupo, Crear Nota en Programa Desarrollo, Modificar Nota en Programa Desarrollo, Eliminar Nota en Programa Desarrollo, Imprimir Programa Desarrollo, Ver Resumen de evento, Crear Asistencia, Modificar Asistencia, Eliminar Asistencia, Buscar Asistente, Buscar Tipo de Asistente, Buscar Asistente por Nombres, Código y Número de documento, Ver asistencia, Subir Foto a Evento, Eliminar Foto de evento, Crear Publicación, Eliminar Publicación, Modificar Publicación, Ver Publicación, Crear Mensaje Grupal y Crear Mensaje de participación (Correo).

Iteración 5: abarca la creación de subgrupos, registro de estudiantes, control de estudios bíblicos, registro de metas por indicadores y creación de parentescos. Los casos de uso implementados son: Crear Sub Grupo, Modificar Sub Grupo, Eliminar Sub Grupo, Ver Sub Grupo, Asignar Integrante a Sub grupo, Eliminar Integrante de Sub Grupo, Buscar Estudiante de la Biblia, Buscar Curso Bíblico, Modificar Estudiante de la Biblia, Eliminar Estudiante de la Biblia, Crear Estudio Bíblico, Buscar Lugar para Estudio Bíblico, Eliminar Estudio Bíblico de Estudiante de la Biblia, Modificar Estudio Bíblico de estudiante de la Biblia, Crear Meta por Indicador del grupo, Modificar Meta por Indicador del grupo, Eliminar Meta por Indicador del grupo, Crear Parentesco, Modificar Parentesco, Eliminar Parentesco, Ver Parentesco, Buscar Tipo de relación por género, Crear Incidencia, Modificar Incidencia, Eliminar Incidencia, Ver Incidencia, Asignar Evidencia a Incidencia, Eliminar Evidencia de incidencia y Ver Evidencias en Incidencia.

Iteración 6: abarca el módulo de seguridad de la aplicación conformado por el registro de usuarios, perfiles, módulos y permisos a grupos. Abarca los siguientes casos de uso: Crear Perfil del sistema, Modificar Perfil, Eliminar Perfil, Ver Perfil, Crear Módulo, Modificar Módulo, Eliminar Módulo, Ver Módulo, Asignar Permiso a Perfil, Eliminar Permiso de perfil, Ver Permiso, Crear Usuario, Modificar Usuario, Eliminar Usuario y Ver usuario.

Iteración 7: esta iteración permite la elaboración de reportes para gestión de los grupos organizados, ayudando a la toma de decisiones del grupo y de la iglesia. Abarca los siguientes reportes: Grupos Registrados, Cumpleaños, Gráfico por Entidad Académica, Gráfico por Religión, Grupos e Integrantes, Asistencia a los Grupos, Estadísticas Generales por: Entidad Académica, Tipo de persona, por Religión, por Género, Ficha por Persona, Análisis de asistencias, Para MiPES: Parejas Misioneras, estudios bíblicos, Asistencia a Escuela Misionera, Termómetro de Escuela Sabática, Monitoreo General, Monitoreo mi Grupo, Reportes de gestión: Líderes con más Eventos Registrados, Personas con más participaciones, Personas con más asistencias, Personas con más faltas, Personas con más faltas justificadas, Instructores bíblicos, Buscar Dones y Talentos, Personas con mejor evaluación, Personas con baja evaluación, Evaluación por entidad académica, % de Asistencia por entidad académica, Lista de personas por número de asistencias, Entrega de materiales y Personas no inscritas.

RESULTADOS

La solución tecnológica ha sido desarrollada de acuerdo a los criterios de planificación de la investigación, se han dado solución a los cambios o actualizaciones solicitadas a través de las reuniones sostenidas con el personal involucrado, se capacitó con respecto al uso del sistema en los laboratorios de sistemas de la institución. A continuación, presentamos algunos resultados dados en figuras del sistema.

1. Página inicial: el usuario ingresa a la página inicial del sistema, puede acceder a la información básica de los grupos registrados y sus integrantes, lista de publicaciones en fotos de las actividades y reuniones de los grupos, lista de los lugares de reunión de los grupos en Google Maps, listado de distritos misioneros y capellanes de facultad, información de la iglesia y coordinadores de grupos.

Figura 9. Página inicial del sistema

2. Iniciar sesión: al ingresar al sistema se visualiza un formulario en el cual el usuario pueda ingresar su nombre de usuario y contraseña, una vez validado pueda realizar las operaciones de acuerdo al rol que posee. Al mismo tiempo

la ventana de inicio de sesión permite consultar a qué grupo pertenece, solo basta ingresar su código universitario o número de DNI, también permite crear un usuario automáticamente, la condición es que la persona debe estar registrada en el sistema, al crear su usuario el sistema envía automáticamente su contraseña a su correo electrónico y también permite restaurar su contraseña en caso de haberse olvidado.

Figura 10. Página iniciar sesión

3. Menú principal: una vez ingresado al sistema, se presenta en la parte superior un menú de opciones organizadas por sus módulos correspondientes, en la parte central se presenta los datos del grupos tales como el nombre, nombre de los líderes y los ideales del grupo, en el lado izquierdo se pueden ver otras opciones como el chat del grupo, solicitudes, lista de cumpleaños del día.

Figura 11. Menú principal del sistema.

4. Registro de integrantes: la opción registro de integrantes se encuentra en el módulo Administrar. Permite visualizar la lista de integrantes del grupo organizada con sus datos, se hace la inscripción de nuevos integrantes, la actualización de sus datos y la eliminación respectiva. Esta es una de las funcionalidades muy importantes ya que, gracias a ello, se puede conseguir organizar bien al grupo y poder gestionarlo.

Figura 12. Gestión de integrantes del grupo.

5. Ficha del integrante: la ficha del integrante es un reporte de las asistencias por cada integrante, lista de participaciones y las incidencias, ya sean positivas o negativas.

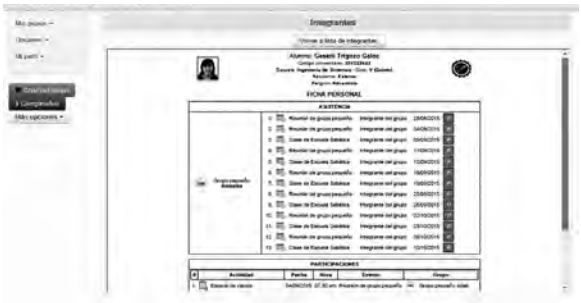

Figura 13. Reporte de la ficha del integrante.

6. Registro de eventos: permite el registro de los eventos que realizan los grupos organizados, se selecciona el tipo de evento, título, fecha y hora, lugar, resumen, descripción. Cada evento permite registrar el programa a desarrollar el control de la asistencia y la evaluación, de acuerdo al tipo de evento.

Figura 14. Registro de eventos por grupo.

7. Registro de asistencia y evaluación: permite registrar la asistencia de los integrantes y otras personas al evento. Existe estado de asistencia tales como: puntual, tardanzas, faltas y faltas justificadas.

Figura 15. Registro de asistencia al evento.

8. Reporte de asistencia por fecha: permite un reporte de asistencia a los grupos filtrado por entidad académica y seleccionando la fecha del evento. Se pueden visualizar los grupos que registraron la asistencia, las personas que asistieron, faltaron y motivos. Este informe ayuda a los capellanes para ayudar a los jóvenes que están pasando por dificultades y así orar por ellos, porque para Dios nada es imposible.

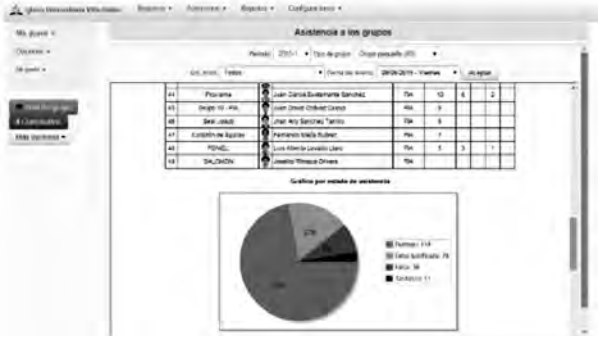

Figura 16. Asistencia a los grupos por fecha.

9. Termómetro de Escuela Sabática: la Iglesia Universitaria Villa Unión trabaja con una tabla comparativa, ahora llamado termómetro, lo que permite evaluar a cada grupo, es por ello que este sistema busca ese fin, ya que se construyó este sistema para capturar esos indicadores también.

Figura 17. Termómetro de la Escuela Sabática.

Se crearon muchos más reportes y formularios para la gestión de los grupos organizados, entre los cuales cumplen con los requisitos del sistema en mención.

CONCLUSIONES

En las primeras fases de este proyecto se logró conocer la problemática existente, resultado de la carencia de la información de los integrantes de los grupos categorizados por: alumnos, docentes, personal general, hermanos (as), visitantes, invitados, egresados, niños (as) que permite la elaboración de informes de trabajo, la gestión de los grupos organizados y el proceso de evaluación. El diagnóstico situacional de los grupos organizados permitió conocer la dinámica organizacional que, a su vez, trajo como resultado una lista de requerimientos que llegaron a formar parte de las características del sistema de información. El análisis de los requerimientos dio como resultado un buen diseño de base de datos, el cual es la base para construir un sistema de calidad que satisfaga eficazmente las necesidades de los clientes.

La aplicación de metodologías ágiles como OpenUP, para el desarrollo de proyectos de software, permite entregar el producto funcional en menor tiempo y de calidad, medido por la satisfacción de los clientes. La implementación por medio de fases y actividades traen como resultados documentos como: visión, arquitectura del sistema de base de datos, manuales de usuario y manual de instalación son claves para obtener un producto de calidad. Además, las fases están bien definidas para poder aplicarlas, y la construcción de una forma iteractiva permite en cada versión entregar software funcional previamente probado.

RECOMENDACIONES

Se recomienda trabajar con los expertos del tema para la definición de indicadores fiables, los cuales midan el desempeño de cada integrante de grupo y del grupo mismo. Esto ayudaría a cada persona trabajar en su mejoramiento personal, tales como puntualidad, participación, comportamiento y espiritualidad. En tal sentido, como trabajo futuro, se tendría que implementar un cuadro de mando para la toma de decisiones en tiempo real.

Mejorar el diseño para dispositivos móviles. Adaptar los procesos de registro de integrantes y control de asistencia a reuniones, o hacer uso del nuevo framework de diseño web Materialize implementado por Google, ya que este se adapta muy bien a dispositivos móviles. Actualizar el framework "Bosdtrad" a "Materialize" implementado por Google para dar una mejor interoperabilidad en el uso del sistema.

Implementar la funcionalidad para evaluar cada actividad de los participantes, midiendo así el grado de responsabilidad. También se puede añadir puntajes para que cada participante se sienta motivado hacia una mejora continua. Además elaborar más reportes que permitan la toma de decisiones para la mejora de cada integrante. Implementar un chat interactivo para mantenerse comunicados constantemente y un mecanismo para compartir materiales, basados en archivos entre líderes de la iglesia y los grupos organizados.

Referencias

Acosta, M. de L. (2013). Estudio de patrones de diseño en plataforma java enterprise edition versión 6 para el desarrollo de aplicaciones web. Ibarra, Ecuador. Retrieved from http://repositorio.utn.edu.ec/bitstream/123456789/2787/1/04 ISC 266 TESIS.pdf

Dhir, S., & Sarraf, S. (2016). Crime and criminal tracking networks & systems using agile methodology. *BIJIT - BVICAM's International Journal of Information Technology,* 8(1), 930–933.

Issaoui, I., Bouassida, N., & Ben-abdallah, H. (2016). Predicting the existence of design patterns based on semantics and metrics. *The International Arab Journal of Information Technology,* 13(2), 310–319.

Kamthan, P. (2013). On the role of Wiki for managing knowledge in agile software development. *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013*, 622–623. http://doi.org/10.1109/CTS.2013.6567299

Khan, J. A., Rehman, I. U., Khan, Y. H., Shah, S. A., & Khan, W. (2016). *Enhancement in agile methodologies using requirement, 28*(2), 1525–1533.

Loniewski, G. (2011). OpenUP/MDRE: A Model-Driven Requirements Engineering Approach for Health-Care Systems. Universidad Politécnica de Valencia. Retrieved from https://riunet.upv.es/bitstream/handle/10251/11652/*Tesina Master* Grzegorz Loniewski.pdf?sequence=1

Qureshi, M. R. J., & Sabir, F. (2013). A comparison of model view controller and model view presenter. *Science International, 25*(1), 7–9. Retrieved from http://arxiv.org/abs/1408.5786

Rao, T. V. N., Hiranmayi, D., Priya, G. S., & Ravi, G. (2014). Assessment of manifestation of agile methodology and computing. *International Journal of Computer Science Engineering and Technology, 4*(12), 358–363.

Ríos, S., Hinojosa, C., & Delgado, R. (2013). *Aplicación de la metodologia openup en el desarrollo del sistema de difusión de gestión del conocimiento de la espe,* 10. Retrieved from http://repositorio.espe.edu.ec/bitstream/21000/6316/1/AC-SISTEMAS-ESPE-047042.pdf

Suárez, J. M., & Gutiérrez, L. E. (2016). Tipificación de dominios de requerimientos para la aplicación de patrones arquitectónicos. *Información Tecnológica, 27*(4), 193–202. http://doi.org/10.4067/S0718-07642016000400021

Ting, D. H., Hsueh, N. L., Yang, C. T., & Chang, C. H. (2015). A cloud service implementation for evaluating design pattern in software evolution. *Journal of Information Science and Engineering, 31*(3), 1051–1070.

Xu, Y., & Liang, P. (2014). Automated software architectural synthesis using patterns: A cooperative coevolution approach. *Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering, 24*(10), 1387–1411. http://doi.org/10.1142/S0218194014400130