UNIVERSIDAD NACIONAL DEL ALTIPLANO
FACULTAD DE INGENIERIA ESTADÍSTICA E INFORMÁTICA
Escuela Profesional de Ingeniería Estadística e Informática

TESIS

SISTEMA PARA LA ADMINISTRACION ACADEMICA DEL CENTRO PREUNIVERSITARIO DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO – PUNO 2015

Presentada por:

Bach. EFRAIN ANCACHI LIMACHI

Para optar el Título Profesional de:

INGENIERO ESTADÍSTICO E INFORMÁTICO

Puno – Perú
2018

UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERIA ESTADISTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

SISTEMA PARA LA ADMINISTRACION ACADEMICA DEL CENTRO PREUNIVERSITARIO DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO – PUNO 2015

TESIS PRESENTADA POR:

Bach. EFRAIN ANCACHI LIMACHI

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ESTADÍSTICO E INFORMÁTICO.

APROBADA POR:

PRESIDENTE :

Dr. Percy Huata Panca

PRIMER MIEMBRO :

M. Sc. Fredy H. Villasante Saravia

SEGUNDO MIEMBRO :

Ing. Alcides Ramos Calcina

DIRECTOR DE TESIS :

Dr. Edgar E. Carpio Vargas

ÁREA	:	Informática
TEMA	:	Sistemas de información
FECHA DE SUSTENTACIÓN	:	06/04/2018

DEDICATORIA

A Braulia, mi madre, quien a pesar de habernos dejado, no ha apartado la mirada de sus hijos.

A Alipio, mi padre, quien a su manera nos ha preparado para afrontar la vida.

A Yessica y Abigail, mi familia quienes a pesar de los malos momentos siempre han estado ahí para darme un “empujoncito”.

A Sandro y Adalidt, con los que he compartido y comparto momentos que solo se pueden vivir con los hermanos.

AGRADECIMIENTOS

Agradezco a los docentes de la Escuela Profesional de Ingeniería Estadística e Informática por haberme acompañado y guiado a lo largo de mi carrera y por brindarme una vida llena de aprendizajes, experiencias y sobre todo amistad.

Le doy gracias a mis padres Alipio y Braulia por haberme apoyado en todo momento, por los valores que me han inculcado y por haberme dado la oportunidad de tener una excelente educación y por sobre todo ser un excelente ejemplo.

A mis hermanos por ser parte importante de mi vida y representar la unidad familiar.
A Sandro y Adalidt por ser un ejemplo de desarrollo profesional a seguir y por llenar mi vida de alegrías y carcajadas cuando más lo he necesitado.

A Fredy, Samuel, Percy, Edgar que más que docentes han sido verdaderos amigos.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

Se plantea como problema la demora en los procesos que conlleva el desarrollo administrativo de cada ciclo formativo del CEPREUNA, esto a causa de la mala implementación de sus sistemas, el presente trabajo, titulado: SISTEMA PARA LA ADMINISTRACION ACADÉMICA DEL CENTRO PREUNIVERSITARIO DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO – PUNO 2015, tiene como objetivo principal: Desarrollar un sistema para la administración académica del Centro de estudios Preuniversitarios de la Universidad Nacional del Altiplano Puno. La metodología usada fue XP por ser flexible, de fácil implementación y fiable, se hizo uso también del Lenguaje Unificado de Modelado para el diseño de los procesos, PHP para la programación back-end a través de CakePHP un framework robusto, confiable y con un amplio soporte por parte de la comunidad de desarrolladores y finalmente se aplicaron las métricas de validación de software según ISO 9126. Finalmente se analizó, diseño e implemento el sistema que mejoró significativamente la administración académica del centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano Puno y ha gozado de una gran aceptación mejorando y agilizando los procesos que conllevan el desarrollo de cada ciclo formativo en esta dependencia.

Palabras clave: Sistema, académico, administración, CakePhp, CEPREUNA.

ABSTRACT

The delay in the processes involved in the administrative development of each training cycle of CEPREUNA is posed as a problem, due to the poor implementation of its systems, the present work, entitled: SISTEMA PARA LA ADMINISTRACION ACADÉMICA DEL CENTRO PREUNIVERSITARIO DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO – PUNO 2015, has as main objective: To develop a system for the academic administration of the Pre-University Studies Center of the National University of Puno Altiplano. The methodology used was XP because it was flexible, easy to implement and reliable, it also made use of the Unified Modeling Language for the design of the processes, PHP for the back-end programming through CakePHP, a robust, reliable framework with a broad support from the developer community and finally software validation metrics according to ISO 9126 were applied. Finally, the system that significantly improved the academic administration of the Pre-university Studies Center of the National University of the Puno Highlands was analyzed, designed and implemented. and has enjoyed great acceptance by improving and streamlining the processes that lead to the development of each training cycle in this unit.

Keywords: System, academic, administration, CakePhp, CEPREUNA.

CAPÍTULO I

PLAN DE INVESTIGACIÓN

1.1. EL PROBLEMA

Para el desarrollo de los ciclos formativos del CEPREUNA, esta dependencia carece de un sistema integrado que facilite el manejo, procesamiento y administración de información, esto a su vez acarrea demoras en todos los procesos que se desarrollan durante cada ciclo, además los malestares generados a docentes y estudiantes por la mala accesibilidad a la información.

La labor administrativa comienza con la planificación del desarrollo de cada ciclo formativo, debiéndose establecer las fechas de inicio, final, periodos de inscripción, estimaciones, etc., también iniciado el proceso de inscripción y registro de pagos para docentes y estudiantes esta actividad se torna dificultosa cuando la concurrencia de alumnos y docentes supera la capacidad de atención del personal a cargo.

Iniciado el ciclo formativo, se inician también los procesos de conteo de forma que para los estudiantes se contabilicen y acumulen asistencias para al final del ciclo hacerse acreedores del puntaje adicional, registro de pagos para

obtener los documentos que lo habilitarían para rendir el examen de admisión modalidad CEPREUNA y para los docentes se registren sus asistencias con el fin de contabilizar el total del sus horas dictadas y poder obtener su remuneración respectiva.

Para ambos casos la recolección, almacenamiento y procesamiento de información es una labor administrativa complicada ya que sin los procedimientos, la mala organización y la carencia de las herramientas adecuadas pueden dar como resultado un ciclo formativo con pocos beneficios económicos, malas experiencias de usuarios (docentes y estudiantes) y actividades retrasadas que afectarían directamente a los procesos dependientes de esta.

1.2. JUSTIFICACION

La Universidad Nacional del Altiplano entre sus modalidades de ingreso cuenta con el Centro de Estudios Preuniversitarios (CEPREUNA) que a diferencia de la modalidad de ingreso por Examen General, esta prepara académicamente por 8 semanas a los postulantes para poder rendir el examen de admisión. Dicha modalidad de ingreso hace necesaria el funcionamiento del CEPREUNA como una dependencia organizacional dentro de la misma universidad.

La labor administrativa que realiza el CEPREUNA, a lo largo de cada ciclo formativo, sin la herramienta adecuada conduce a un largo y poco productivo periodo de procesamiento de datos todo con el fin de obtener información consolidada para el final de cada ciclo formativo.

En la actualidad, las tecnologías de información y comunicación (TIC) están cambiando las formas de acceso a la información, los modos de comunicación y la manera de relacionarnos a tal punto que la generación, procesamiento, y transformación de la información se está convirtiendo en un factor de poder. La productividad y la competitividad dependen cada vez más de la capacidad de generar y aplicar información.

La metodología de desarrollo y la aplicación de las métricas de evaluación de calidad de software, determinarán si este puede pasar a ser una herramienta adecuada para el manejo y administración de información en el CEPREUNA.

1.3. OBJETIVOS

OBJETIVO GENERAL:

Desarrollar un sistema para la administración académica del Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano – Puno 2015.

OBJETIVOS ESPECÍFICOS:

- Analizar los requerimientos del Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano – Puno.
- Diseñar e implementar un sistema web basado en la metodología XP utilizando CakePHP.
- Evaluar la aceptación del software haciendo uso de las métricas establecidas en la ISO 9126.

1.4. HIPÓTESIS GENERAL

La puesta en marcha de un sistema administrativo mejora significativamente la administración académica del Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano – Puno.

CAPÍTULO II

MARCO TEÓRICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Romero (2013), Se consiguió implementar una solución automatizada capaz de administrar los programas educativos, planes y tareas de los alumnos del centros de educación especial junto con otros procesos en gestión educativa, asimismo, los esfuerzos y tiempo invertidos en el análisis y diseño de la solución posibilitaron la cobertura de todos los requerimientos funcionales del usuario maximizando las funcionalidades deseadas del producto enriqueciéndolas con aportes provenientes de otros sistemas, también concluye que la arquitectura en capas ofrece una mejor escalabilidad para futuras integraciones con nuevas herramientas y servicios aplicando la reutilización de componentes.

Soto (2014), El modelo de sistema basado en XP permite satisfacer los requerimientos determinados en un contexto de gestión Municipal. Como resultado de la etapa de investigación del funcionamiento interno de la municipalidad, se planteó un modelo general para realizar un trámite o servicio Municipal, que sirvió de entrada para el flujo de trabajo en el prototipo.

Cáceres Colchado (2014), Se logró aplicar una metodología de desarrollo de software que logró planificar y controlar los Proyectos de Software de la Unidad de Informática, estableciendo fecha para los entregables y proporcionando un calendario establecido para sus actividades, también el contar con el Sistema de Gestión de Calidad y poseer una definición formal de los procesos y procedimientos al inicio del proyecto y a su vez mantener reuniones de planificación y revisión del Sprint con los usuarios permitió definir correctamente sus requerimientos y plasmar estos dentro del Sistema Web de Admisión.

Huanca (2015), Se ha desarrollado el Sistema de Información para la Administración de Programas Sociales (SIAPS), mejorando la gestión de la información en la Gerencia de Desarrollo Social de la Municipalidad Provincial de Azángaro, permitiendo realizar los diferentes procesos con mayor control, reducción de costos y tiempo con un alto grado de productividad.

Domínguez (2016), La percepción subjetiva y según los resultados obtenidos, los usuarios se sintieron satisfechos empleando el software de Gestión de Matricula proporcionado por la unidad de desarrollo del Sistema Único de Matricula, por encima del nivel esperado. Esto revela aspectos positivos del producto en cuanto a la disposición favorable que genera entre ellos, ya sea porque la mayoría manifestó la decisión de continuar utilizándolo, o porque también indicó que aprendió a usarlo sin un gran esfuerzo y que lo recomendaría a otros usuarios de características semejantes.

Chambilla (2017), Se desarrolló un sistema automatizado para los procesos de formulación y programación de actividades, articuladas a las

acciones estratégicas de la institución, permitiendo consolidar la información de manera automática al ingreso de cada formato, reduciendo este procedimiento de 14 días (2012) a 2 días en la actualidad, también concluye: Se implementó un sistema que sirve de apoyo para la administración de los procesos de la gestión municipal en cada fase de formulación, programación, aprobación, evaluación y ejecución del Plan Operativo Institucional de la Municipalidad Provincial de Arequipa, lo que permitió la mejora de los indicadores de desempeño de la municipalidad de un 40.76% en el año 2012 a un 91.25% en la actualidad.

2.2. BASE TEÓRICA

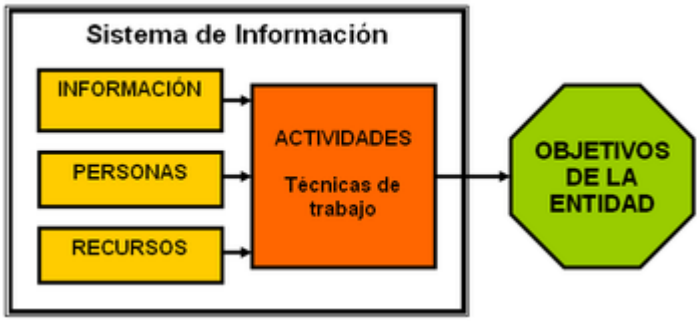
2.1.1. Sistemas de Información

Sistema de información se entiende como el conjunto de tecnologías, procesos, aplicaciones de negocios y software disponibles para las personas dentro de una organización. Un sistema de información debe cumplir con los siguientes componentes básicos interactuando entre sí:

- El hardware, equipo físico utilizado para procesar y almacenar datos,
- El software y los procedimientos utilizados para transformar y extraer información,
- Los datos que representan las actividades de la empresa,
- La red que permite compartir recursos entre computadoras y dispositivos,
- Las personas que desarrollan, mantienen y utilizan el sistema.

Los sistemas de información son una combinación de tres partes principales: las personas, la información y los recursos. (Shaikh, Karjaluoto, 2015).

Figura 1. Sistema de información



Fuente: Wikipedia

2.1.2. Software

Se conoce como software al soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware. La interacción entre el Software y el Hardware hace operativa una computadora (u otro dispositivo), es decir, el Software envía instrucciones que el Hardware ejecuta, haciendo posible su funcionamiento. (Diccionario de la lengua española, 2015).

2.1.3. Sistema de Gestión de Base de Datos (SGDB)

Es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. Los usuarios pueden acceder a la información usando herramientas

específicas de interrogación y de generación de informes, o bien mediante aplicaciones al efecto.

Estos sistemas también proporcionan métodos para mantener la integridad de los datos, para administrar el acceso de usuarios a los datos y para recuperar la información si el sistema se corrompe. Permiten presentar la información de la base de datos en variados formatos. La mayoría incluyen un generador de informes. También pueden incluir un módulo gráfico que permita presentar la información con gráficos y tablas.

Hay muchos tipos distintos según cómo manejen los datos y muchos tamaños distintos de acuerdo a si operan en computadoras personales y con poca memoria o grandes sistemas que funcionan en mainframes con sistemas de almacenamiento especiales.

Generalmente se accede a los datos mediante lenguajes de interrogación, lenguajes de alto nivel que simplifican la tarea de construir las aplicaciones. También simplifican la interrogación y la presentación de la información. Un SGBD permite controlar el acceso a los datos, asegurar su integridad, gestionar el acceso concurrente a ellos, recuperar los datos tras un fallo del sistema y hacer copias de seguridad. Las bases de datos y los sistemas para su gestión son esenciales para cualquier área de negocio, y deben ser gestionados con esmero. (Silberchatz, 2014).

2.1.4. Programación extrema (XP)

eXtreme Programming (XP) surge como una nueva manera de encarar proyectos de software, proponiendo una metodología basada esencialmente

en la simplicidad y agilidad.

Las metodologías de desarrollo de software tradicionales (ciclo de vida en cascada, evolutivo, en espiral, iterativo, etc.) aparecen, comparados con los nuevos métodos propuestos en XP, como pesados y poco eficientes. La crítica más frecuente a estas metodologías “clásicas” es que son demasiado burocráticas. Hay tanto que hacer para seguir la metodología que, a veces, el ritmo entero del desarrollo se retarda. Como respuesta a esto, se ha visto en los últimos tiempos el surgimiento de “Metodologías Ágiles”. Estos nuevos métodos buscan un punto medio entre la ausencia de procesos y el abuso de los mismos, proponiendo un proceso cuyo esfuerzo valga la pena.

Los métodos ágiles cambian significativamente algunos de los énfasis de las metodologías “clásicas”: (Folwer, 2015)

- Los métodos ágiles son adaptables en lugar de predictivos. Los métodos “clásicos” tienden a intentar planear una gran parte del proceso del software en gran detalle para un plazo largo de tiempo. Esto funciona bien hasta que las cosas cambian. Así que su naturaleza es resistirse al cambio. Para los métodos ágiles, no obstante, el cambio es bienvenido. Intentan ser procesos que se adaptan y crecen en el cambio.
- Los métodos ágiles son orientados a la gente y no orientados al proceso. El objetivo de los métodos “clásicos” es definir un proceso que funcionará bien independientemente de quien lo utilice. Los métodos ágiles afirman que ningún proceso podrá nunca maquillar las habilidades del equipo de desarrollo, de modo que el papel del proceso es apoyar al equipo de desarrollo en su trabajo.

XP es una de las llamadas metodologías ágiles de desarrollo de software más exitosas de los tiempos recientes. La metodología propuesta en XP está diseñada para entregar el software que los clientes necesitan en el momento en que lo necesitan. XP alienta a los desarrolladores a responder a los requerimientos cambiantes de los clientes, aún en fases tardías del ciclo de vida del desarrollo.

La metodología también enfatiza el trabajo en equipo. Tanto gerentes como clientes y desarrolladores son partes del mismo equipo dedicado a entregar software de calidad.

Ciclo de vida del software XP

Para apreciar los conceptos del ciclo de desarrollo de software en XP introduciremos brevemente los conceptos principales de las metodologías de desarrollo de software tradicionales. (Weitzenfeld, 2015).

- **Modelo en cascada:** El modelo de cascada tiene sus orígenes en la década de 1970, y se define como una secuencia de actividades bien planificadas y estructuradas. El proceso distingue claramente las fases de especificación de las de desarrollo y éstas, a su vez, de las de testing. Es, seguramente, la metodología más extendida y utilizada. Este modelo se basa fuertemente en que cada detalle de los requisitos se conoce de antemano, previo de comenzar la fase de codificación o desarrollo, y asume, además, que no existirán cambios significativos en los mismos a lo largo del ciclo de vida del desarrollo. (Casallas, 2015).
- **Modelo Incremental:** El modelo incremental consiste en un desarrollo inicial de la arquitectura completa del sistema, seguido de sucesivos

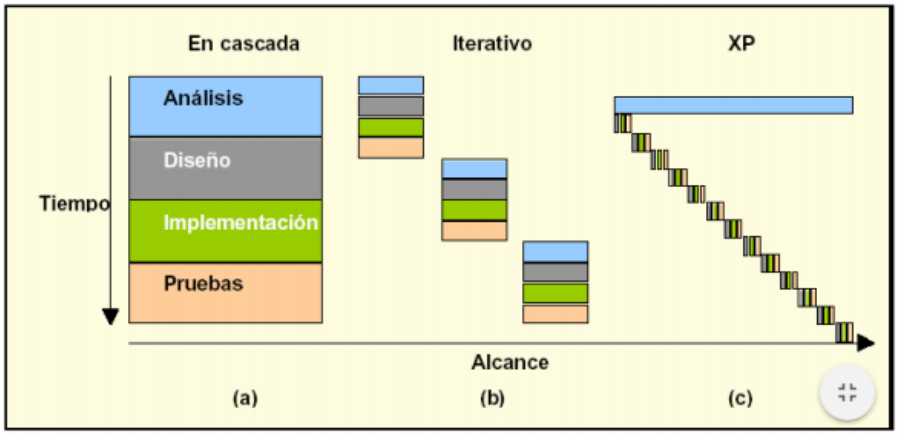
incrementos funcionales. Cada incremento tiene su propio ciclo de vida y se basa en el anterior, sin cambiar su funcionalidad ni sus interfaces. Una vez entregado un incremento, no se realizan cambios sobre el mismo, sino únicamente corrección de errores. Dado que la arquitectura completa se desarrolla en la etapa inicial, es necesario, al igual que en el modelo en cascada, conocer los requerimientos completos al comienzo del desarrollo. Respecto al modelo en cascada, el incremental tiene la ventaja de entregar una funcionalidad inicial en menor tiempo.

- **Modelo Evolutivo:** El modelo evolutivo es, en cierta forma, similar al incremental, pero admite que la especificación no esté completamente determinada al comienzo del ciclo de vida. Los requerimientos que estén suficientemente detallados al comienzo darán lugar a una entrega inicial, mientras que los siguientes incrementos serán cambios progresivos que implementen “deltas” de especificación de requerimientos. El modelo admite que, si la especificación no es suficientemente clara al principio, puede desarrollarse un prototipo experimental, que tiene como función validar o identificar los requisitos del sistema.
- **Modelo espiral:** El modelo de espiral, introducido por Barry Bohem a fines de la década de 1980, intenta combinar las ventajas del modelo en cascada con el modelo evolutivo. El modelo enfatiza el estudio de los riesgos del proyecto, como por ejemplo las especificaciones incompletas. Se prevé, en este modelo, varios ciclos o “vueltas de espiral”, cada uno de ellos con cuatro etapas: Definición de objetivos, Evaluación y reducción del riesgo, Desarrollo y validación y Planificación del siguiente ciclo. En este modelo, una actividad comienza solo cuando se entienden los objetivos y riesgos

involucrados. El desarrollo se incrementa en cada etapa, generando una solución completa. La metodología en espiral ha sido utilizado con éxito en grandes sistemas, pero su complejidad la hace desaconsejable para el desarrollo de sistemas medianos o pequeños. (Casallas, 2015)

- **Modelo XP:** El ciclo de vida de un proyecto XP incluye, al igual que las otras metodologías, entender lo que el cliente necesita, estimar el esfuerzo, crear la solución y entregar el producto final al cliente. Sin embargo, XP propone un ciclo de vida dinámico, donde se admite expresamente que, en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto. Por esto, se trata de realizar ciclos de desarrollo cortos (llamados iteraciones), con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas y prácticas que caracterizan a XP (y que serán detalladas más adelante).

Figura 2. Ciclos de vida del software



Fuente: Wikipedia.

Si bien el ciclo de vida de un proyecto XP es muy dinámico, se puede separar en fases. Varios de los detalles acerca de las tareas de éstas fases se detallan a continuación:

- **Fase de exploración:** Es la fase en la que se define el alcance general del proyecto. En esta fase, el cliente define lo que necesita mediante la redacción de sencillas “historias de usuarios”. Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración. Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.
- **Fase de planificación:** La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas, “Release Plan”.
- **Fase de iteraciones:** Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. El cliente, por lo tanto,

también debe participar activamente durante esta fase del ciclo. Las iteraciones son también utilizadas para medir el progreso del proyecto. Una iteración terminada sin errores es una medida clara de avance.

- **Fase de puesta en producción:** Si bien al final de cada iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente no poner el sistema en producción hasta tanto no se tenga la funcionalidad completa. En esta fase no se realizan más desarrollos funcionales, pero pueden ser necesarias tareas de ajuste (“fine tuning”).

(Casallas, 2015).

Reglas y practicas XP

La metodología XP tiene un conjunto importante de reglas y prácticas. En forma genérica, se pueden agrupar en: (Wells, 2013).

- **Planificación:** La metodología XP plantea la planificación como un dialogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. El proyecto comienza recopilando “Historias de usuarios”, las que sustituyen a los tradicionales “casos de uso”. Una vez obtenidas las “historias de usuarios”, los programadores evalúan rápidamente el tiempo de desarrollo de cada una. Si alguna de ellas tiene “riesgos” que no permiten establecer con certeza la complejidad del desarrollo, se realizan pequeños programas de prueba (“spikes”), para reducir estos riesgos. Una vez realizadas estas estimaciones, se organiza una reunión de planificación, con los diversos actores del proyecto (cliente, desarrolladores, gerentes), a los efectos de establecer un plan o cronograma de entregas (“Release Plan”) en los que

todos estén de acuerdo. Una vez acordado este cronograma, comienza una fase de iteraciones, en dónde en cada una de ellas se desarrolla, prueba e instala unas pocas “historias de usuarios”. Según Martín Fowler (uno de los firmantes del “Agile Manifesto”), los planes en XP se diferencian de las metodologías tradicionales en tres aspectos:

- Simplicidad del plan. No se espera que un plan requiera de un “gurú” con complicados sistemas de gerenciamiento de proyectos.
- Los planes son realizados por las mismas personas que realizarán el trabajo.
- Los planes no son predicciones del futuro, sino simplemente la mejor estimación de cómo saldrán las cosas. Los planes son útiles, pero necesitan ser cambiados cuando las circunstancias lo requieren. De otra manera, se termina en situaciones en las que el plan y la realidad no coinciden, y en estos casos, el plan es totalmente inútil.

Los conceptos básicos de esta planificación son los siguientes:

- **Historias de usuarios:** Las “Historias de usuarios” (“User stories”) sustituyen a los documentos de especificación funcional, y a los “casos de uso”. Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento

de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios. Las historias de usuarios deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia.

- **Plan de entregas:** El cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). XP denomina a esta reunión "Juego de planeamiento" ("Planning game"), pero puede denominarse de la manera que sea más apropiada al tipo de empresa y cliente (por ejemplo, Reunión de planeamiento, "Planning meeting" o "Planning workshop") Típicamente el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores. Luego de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario.
- **Plan de iteraciones:** Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada

historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores. Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir.

- **Reuniones diarias de seguimiento:** El objetivo de tener reuniones diarias es mantener la comunicación entre el equipo, y compartir problemas y soluciones. En la mayoría de estas reuniones, gran parte de los participantes simplemente escuchan, sin tener mucho que aportar. Para no quitar tiempo innecesario del equipo, se sugiere realizar estas reuniones en círculo y de pie.

- **Diseño:** La metodología XP hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son los siguientes:

  - **Simplicidad:** Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione. Se sugiere nunca adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando.

  - **Soluciones "spike":** Cuando aparecen problemas técnicos, o cuando es difícil de estimar el tiempo para implementar una historia de usuario, pueden utilizarse pequeños programas de prueba (llamados "spike"), para explorar diferentes soluciones. Estos programas son únicamente

para probar o evaluar una solución, y suelen ser desechados luego de su evaluación.

- **Recodificación:** La recodificación ("refactoring") consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de hacerlo más simple, conciso y/o entendible. Muchas veces, al terminar de escribir un código de programa, pensamos que, si lo comenzáramos de nuevo, lo hubiéramos hecho en forma diferente, más clara y eficientemente. Sin embargo, como ya está pronto y "funciona", rara vez es reescrito. Las metodologías de XP sugieren recodificar cada vez que sea necesario. Si bien, puede parecer una pérdida de tiempo innecesaria en el plazo inmediato, los resultados de ésta práctica tienen sus frutos en las siguientes iteraciones, cuando sea necesario ampliar o cambiar la funcionalidad. La filosofía que se persigue es, como ya se mencionó, tratar de mantener el código más simple posible que implemente la funcionalidad deseada.
- **Metáforas:** Una "metáfora" es algo que todos entienden, sin necesidad de mayores explicaciones. La metodología XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, y guiar la estructura y arquitectura del mismo. Por ejemplo, puede ser una guía para la nomenclatura de los métodos y las clases utilizadas en el diseño del código. Tener nombres claros, que no requieran de mayores explicaciones, redunda en un ahorro de tiempo. Es muy importante que el cliente y el grupo de desarrolladores estén de acuerdo y compartan esta "metáfora", para que puedan dialogar en un "mismo idioma". Una buena metáfora debe ser fácil de comprender para el

cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto. Sin embargo, ésta práctica resulta, muchas veces, difícil de realizar. En un trabajo realizado en el School of Computer Science del Carnegie Mellon, se cuestiona la utilidad de su uso. (Tomayko, Herbsleb, 2003).

- **Desarrollo del código**
  - **Disponibilidad del cliente:** Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto con la metodología XP. Al comienzo del proyecto, el cliente debe proporcionar las historias de usuarios. Pero, dado que estas historias son expresamente cortas y de "alto nivel", no contienen los detalles necesarios para realizar el desarrollo del código. Estos detalles deben ser proporcionados por el cliente, y discutidos con los desarrolladores, durante la etapa de desarrollo. No se requieren de largos documentos de especificaciones, sino que los detalles son proporcionados por el cliente, en el momento adecuado, "cara a cara" a los desarrolladores.

    Si bien esto parece demandar del cliente recursos por un tiempo prolongado, debe tenerse en cuenta que en otras metodologías este tiempo es insumido por el cliente en realizar los documentos detallados de especificación. Adicionalmente, al estar el cliente en todo el proceso, puede prevenir a tiempo de situaciones no deseables, o de funcionamientos que no eran los que en realidad se deseaban. En otras

metodologías, estas situaciones son detectadas en forma muy tardía del ciclo de desarrollo, y su corrección puede llegar a ser muy complicada.

- **Uso de estándares:** Si bien esto no es una idea nueva, XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la recodificación.
- **Programación dirigida por las pruebas:** En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de los tests, es usualmente realizada sobre el final del proyecto, o sobre el final del desarrollo de cada módulo. La metodología XP propone un modelo inverso, en el que, lo primero que se escribe son los test que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas. Las pruebas a los que se refiere esta práctica, son las pruebas unitarias, realizados por los desarrolladores. La definición de estos test al comienzo, condiciona o "dirige" el desarrollo.
- **Programación en pares:** XP propone que se desarrolle en pares de programadores, ambos trabajando juntos en un mismo ordenador. Si bien parece que ésta práctica duplica el tiempo asignado al proyecto (y por ende, los costos en recursos humanos), al trabajar en pares se minimizan los errores y se logran mejores diseños, compensando la inversión en horas. El producto obtenido es por lo general de mejor calidad que cuando el desarrollo se realiza por programadores individuales. En un estudio realizado por Cockburn y Williams (Cockburn, Williams, 2000), se concluye que la programación en pares tiene un sobre costo aproximado de 15%, y no de un 100% como se puede

pensar a priori. Este sobre costo es rápidamente pagado por la mejor calidad obtenida en el producto final. Adicionalmente, la programación en pares tiene las siguientes ventajas:

- La mayoría de los errores se descubren en el momento en que se codifican, ya que el código es permanentemente revisado por dos personas.
- La cantidad de defectos encontrados en las pruebas es estadísticamente menor.
- Los diseños son mejores y el código más corto.
- El equipo resuelve problemas en forma más rápida.
- Las personas aprenden significativamente más, acerca del sistema y acerca de desarrollo de software.
- El proyecto termina con más personas que conocen los detallas de cada parte del código.
- Las personas aprenden a trabajar juntas, generando mejor dinámica de grupo y haciendo que la información fluya rápidamente.
- Las personas disfrutan más de su trabajo.

- **Integraciones permanentes:** Todos los desarrolladores necesitan trabajar siempre con la "última versión". Realizar cambios o mejoras sobre versiones antiguas causan graves problemas, y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas. Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez.

- **Propiedad colectiva del código:** En un proyecto XP, todo el equipo puede contribuir con nuevas ideas que apliquen a cualquier parte del proyecto. Asimismo, cualquier pareja de programadores puede cambiar el código que sea necesario para corregir problemas, agregar funciones o recodificar. En otras metodologías, este concepto puede parecer extraño. Muchas veces se asume que, si hay algo de propiedad colectiva, la responsabilidad también es colectiva. Y que "todos sean responsables", muchas veces significa que "nadie es responsable". Ward Cunningham explica en una entrevista con Bill Venners (Venners, 2003), que este razonamiento no es correcto cuando se trabaja con la metodología de XP. En este caso, quienes encuentran un problema, o necesitan desarrollar una nueva función, pueden resolverlo directamente, sin necesidad de "negociar" con el "dueño" o autor del módulo (ya que, de hecho, este concepto no existe en XP). Muchas veces, explica Cunningham, una solución pasa por la recodificación de varios módulos, que atraviesan de forma horizontal una determinada jerarquía vertical. Si es necesario dialogar y convencer al encargado de cada módulo, posiblemente la solución no se pueda implementar, por lo menos en tiempos razonables. En XP, se promueve la recodificación, en aras de generar códigos más simples y adaptados a las realidades cambiantes. Cualquier pareja de programadores puede tomar la responsabilidad de este cambio. Los testeos permanentes deberían de aseguran que los cambios realizados cumplen con lo requerido, y además, no afectan al resto de las funcionalidades.

- **Ritmo sostenido:** La metodología XP indica que debe llevarse un ritmo sostenido de trabajo. Anteriormente, ésta práctica se denominaba “Semana de 40 horas”. Sin embargo, lo importante no es si se trabajan, 35, 40 o 42 horas por semana. El concepto que se desea establecer con esta práctica es el de planificar el trabajo de manera de mantener un ritmo constante y razonable, sin sobrecargar al equipo. Cuando un proyecto se retrasa, trabajar tiempo extra puede ser más perjudicial que beneficioso. El trabajo extra desmotiva inmediatamente al grupo e impacta en la calidad del producto. En la medida de lo posible, se debería renegociar el plan de entregas (“Release Plan”), realizando una nueva reunión de planificación con el cliente, los desarrolladores y los gerentes. Adicionalmente, agregar más desarrolladores en proyectos ya avanzados no siempre resuelve el problema.

- **Pruebas**

  - **Pruebas unitarias:** Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente, las pruebas deben ser definidas antes de realizar el código (“Test-driven programming”). Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo.

- **Detección y corrección de errores:** Cuando se encuentra un error (“bug”), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto.
- **Pruebas de aceptación:** Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Las pruebas de aceptación son consideradas como “pruebas de caja negra” (“Black box system tests”). Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información.

Valores XP

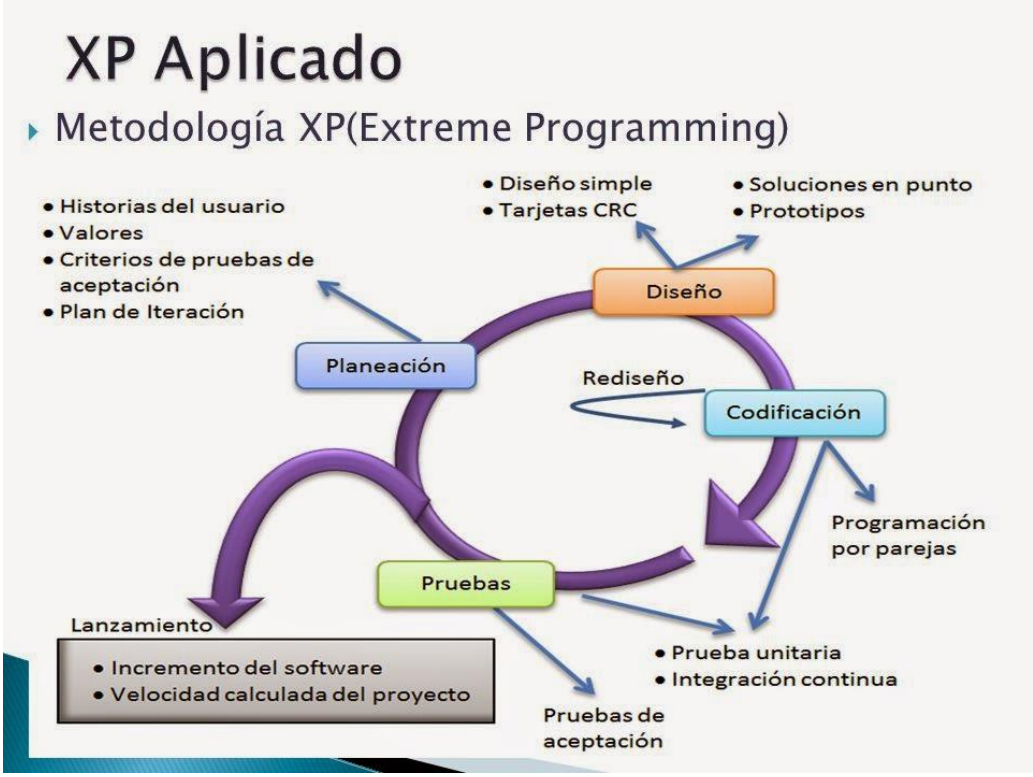
XP se basa en cuatro valores, que deben estar presentes en el equipo de desarrollo para que el proyecto tenga éxito.

- **Comunicación:** Muchos de los problemas que existen en proyectos de software (así como en muchos otros ámbitos) se deben a problemas de comunicación entre las personas. La comunicación permanente es fundamental en XP. Dado que la documentación es escasa, el diálogo

frontal, cara a cara, entre desarrolladores, gerentes y el cliente es el medio básico de comunicación. Una buena comunicación tiene que estar presente durante todo el proyecto.

- **Simplicidad:** XP, como metodología ágil, apuesta a la sencillez, en su máxima expresión. Sencillez en el diseño, en el código, en los procesos, etc. La sencillez es esencial para que todos puedan entender el código, y se trata de mejorar mediante recodificaciones continuas.
- **Retroalimentación:** La retroalimentación debe funcionar en forma permanente. El cliente debe brindar retroalimentación de las funciones desarrolladas, de manera de poder tomar sus comentarios para la próxima iteración, y para comprender, cada vez más, sus necesidades. Los resultados de las pruebas unitarias son también una retroalimentación permanente que tienen los desarrolladores acerca de la calidad de su trabajo.
- **Coraje:** Cuando se encuentran problemas serios en el diseño, o en cualquier otro aspecto, se debe tener el coraje suficiente como para encarar su solución, sin importar que tan difícil sea. Si es necesario cambiar completamente parte del código, hay que hacerlo, sin importar cuanto tiempo se ha invertido previamente en el mismo.

Figura 3: Metodología de desarrollo XP



Fuente:http://wikipedia.org

2.1.5 ISO 9126

Es un estándar internacional para la evaluación de la calidad del software, este estándar está dividido en cuatro partes las cuales dirigen, realidad, métricas externas, métricas internas y calidad en las métricas de uso y expendido. El modelo de calidad establecido en la primera parte del estándar, ISO 9126-1, clasifica la calidad del software en un conjunto estructurado de características y sub características de la siguiente manera:

- **Funcionalidad:** Un conjunto de atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas. Las

funciones son aquellas que satisfacen las necesidades implícitas o explícitas.

- **Adecuación:** Atributos del software relacionados con la presencia y aptitud de un conjunto de funciones para tareas especificadas.
- **Exactitud:** Atributos del software relacionados con la disposición de resultados o efectos correctos o acordados.
- **Interoperabilidad:** Atributos del software que se relacionan con su habilidad para la interacción con sistemas especificados.
- **Seguridad:** Atributos del software relacionados con su habilidad para prevenir acceso no autorizado ya sea accidental o deliberado, a programas y datos.

- **Fiabilidad:** Un conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período establecido.
  - **Madurez:** Atributos del software que se relacionan con la frecuencia de falla por fallas en el software.
  - **Recuperabilidad:** Atributos del software que se relacionan con la capacidad para restablecer su nivel de desempeño y recuperar los datos directamente afectos en caso de falla y en el tiempo y esfuerzo relacionado para ello.
  - **Tolerancia a fallos:** Atributos del software que se relacionan con su habilidad para mantener un nivel especificado de desempeño en casos de fallas de software o de una infracción a su interfaz especificada.

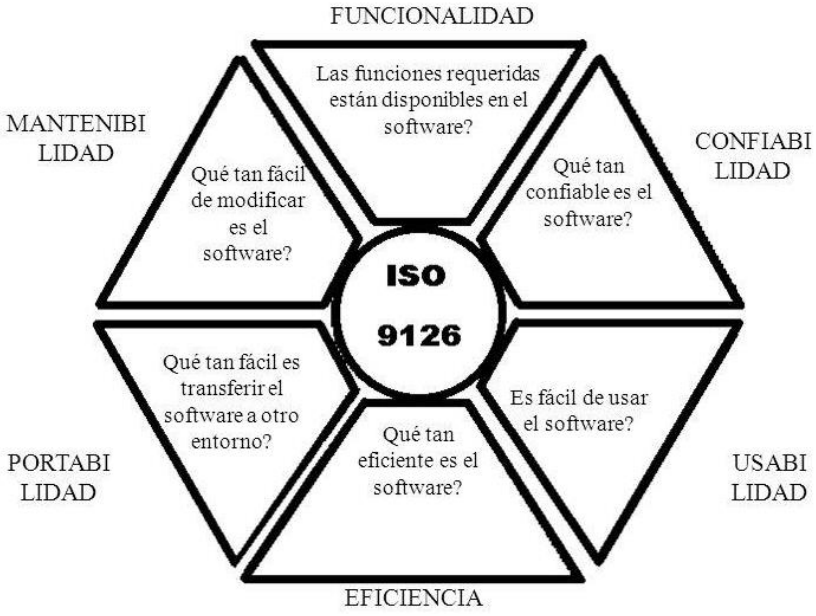
- **Cumplimiento de Fiabilidad:** La capacidad del producto software para adherirse a normas, convenciones o legislación relacionadas con la fiabilidad.

- **Usabilidad:** Un conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios.
  - **Aprendizaje:** Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.
  - **Comprensión:** Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.
  - **Operatividad:** Atributos del software que se relacionan con el esfuerzo del usuario para la operación y control del software.
- **Eficiencia:** Conjunto de atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas.
  - **Comportamiento en el tiempo:** Atributos del software que se relacionan con los tiempos de respuesta y procesamiento y en las tasas de rendimientos en desempeñar su función.
  - **Comportamiento de recursos:** Usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas.
- **Mantenibilidad:** Conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.
  - **Estabilidad:** Atributos del software relacionados con el riesgo de efectos inesperados por modificaciones.

- **Facilidad de análisis:** Atributos del software relacionados con el esfuerzo necesario para el diagnóstico de deficiencias o causas de fallos, o identificaciones de partes a modificar.
- **Facilidad de cambio:** Atributos del software relacionados con el esfuerzo necesario para la modificación, corrección de falla, o cambio de ambiente.
- **Facilidad de pruebas:** Atributos del software relacionados con el esfuerzo necesario para validar el software modificado.

- **Portabilidad:** Conjunto de atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra.
  - Capacidad de instalación: Atributos del software relacionados con el esfuerzo necesario para instalar el software en un ambiente especificado.
  - Capacidad de reemplazamiento: Atributos del software relacionados con la oportunidad y esfuerzo de usar el software en lugar de otro software especificado en el ambiente de dicho software especificado.
- **Calidad en uso:** Conjunto de atributos relacionados con la aceptación por parte del usuario final y Seguridad.
  - Eficacia: Atributos relacionados con la eficacia del software cuando el usuario final realiza los procesos.
  - Productividad: Atributos relacionados con el rendimiento en las tareas cotidiana realizadas por el usuario final.
  - Seguridad: Atributos para medir los niveles de riesgo.
  - Satisfacción: Atributos relacionados con la satisfacción de uso del software.

Figura 4: ISO 916



Fuente: http://csi-sandra-unidad4.blogspot.com

2.3.6. PHP

PHP es un acrónimo recursivo que significa PHP Hypertext Preprocessor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdorf; sin embargo, la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP, al no haber una especificación formal. Publicado con la PHP License, la Free Software Foundation considera esta licencia como software libre.

VISIÓN GENERAL

PHP puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. El lenguaje PHP se encuentra instalado en más de 20 millones de sitios web y en un millón

de servidores. El número de sitios basados en PHP se ha visto reducido progresivamente en los últimos años, con la aparición de nuevas tecnologías como Node.JS, Golang, ASP.NET, etc.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Aunque todo en su diseño está orientado a facilitar la creación de sitios web, es posible crear aplicaciones con una interfaz gráfica para el usuario, utilizando alguna extensión como puede ser PHP-Qt, PHP-GTK,5 WxPHP, WinBinder, Roadsend PHP, Phalanger, Phc o HiP Hop VM. También puede ser usado desde la línea de comandos, de la misma manera como Perl o Python pueden hacerlo; a esta versión de PHP se la llama PHP-CLI (Command Line Interface).

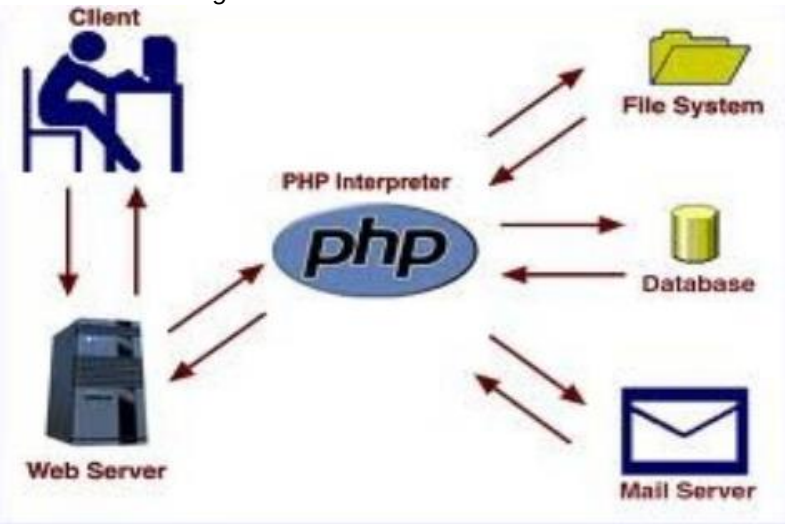
Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos PDF,7 Flash, así como imágenes en diferentes formatos.

Permite la conexión a diferentes tipos de servidores de bases de datos tanto SQL como NoSQL tales como MySQL, PostgreSQL, Oracle, ODBC,

DB2, Microsoft SQL Server, Firebird, SQLite o MongoDB.8

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como Unix (y de ese tipo, como Linux o Mac OS X) y Microsoft Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

Figura 5: Funcionamiento de PHP



Fuente: https://smr.iesharia.org/

Sintaxis

El intérprete de PHP solo ejecuta el código que se encuentra entre sus delimitadores. Los delimitadores más comunes son <?php para abrir una sección PHP y ?> para cerrarla. El propósito de estos delimitadores es separar el código PHP del resto de código, como por ejemplo el HTML. Las variables se prefijan con el símbolo del dólar ($) y no es necesario indicar su tipo. Las variables, a diferencia de las funciones, distinguen entre mayúsculas y minúsculas. Las cadenas de caracteres pueden ser encapsuladas tanto en dobles comillas como en comillas simples, aunque en el caso de las primeras, se pueden insertar variables en la cadena directamente, sin necesidad de concatenación. Los comentarios se pueden escribir bien con dos barras al

principio de la línea, o con una almohadilla. También permite comentarios multi-línea encapsulados en /* */. En cuanto a las palabras clave, PHP comparte con la mayoría de otros lenguajes con sintaxis C las condiciones con *if*, los bucles con *for* y *while* y los retornos de funciones. Como es habitual en este tipo de lenguajes, las sentencias deben acabar con punto y coma (;).

Figura 6: Sintaxis básica PHP

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <title> Ejemplo básico PHP</title>
    </head>
    <body>
        <?php
            echo 'Hola mundo';
        ?>
    </body>
</html>
```

Fuente: Elaboración propia

CARACTERISTICAS

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- Es considerado un lenguaje fácil de aprender, ya que en su desarrollo se simplificaron distintas especificaciones, como es el caso de la definición de las variables primitivas, ejemplo que se hace evidente en el uso de php arrays.
- El código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador.

- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, aun haciéndolo, el programador34 puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.
- Debido a su flexibilidad ha tenido una gran acogida como lenguaje base para las aplicaciones WEB de manejo de contenido, y es su uso principal.

DESVENTAJAS

- Como es un lenguaje que se interpreta en ejecución, para ciertos usos puede resultar un inconveniente que el código fuente no pueda ser ocultado. La ofuscación es una técnica que puede dificultar la lectura del código pero no necesariamente impide que el código sea examinado.
- Debido a que es un lenguaje interpretado, un script en PHP suele funcionar considerablemente más lento que su equivalente en un lenguaje de bajo nivel, sin embargo este inconveniente se puede minimizar con técnicas de caché tanto en archivos como en memoria.
- En las versiones previas a la 7, las variables no son tipificadas, lo cual dificulta a los diferentes IDEs ofrecer asistencias para el tipificado del código, aunque esto no es realmente un inconveniente del lenguaje en sí. Esto es solventado por algunos IDEs añadiendo un comentario con el tipo a la declaración de la variable.

2.3.7. CakePHP

Es un marco de desarrollo (framework) rápido para PHP, libre, de código abierto. Se trata de una estructura que sirve de base a los programadores para que éstos puedan crear aplicaciones Web. Su principal objetivo es poder trabajar de forma estructurada y rápida, sin pérdida de flexibilidad.

Con CakePHP el desarrollo web ya no es monótono porque ofrece las herramientas para empezar a escribir el código que realmente se necesita: la lógica específica de la aplicación.

CakePHP tiene un equipo de desarrolladores y una comunidad activa, lo que añade valor al proyecto. Con CakePHP, además de no tener que reinventar la rueda, el núcleo de la aplicación se mejora constantemente y está bien probado. Esta es una lista breve con las características que ofrece CakePHP:

- Comunidad activa y amistosa
- Licencia flexible
- Compatible con PHP4, PHP5 y PHP7
- CRUD integrado para la interacción con la base de datos
- Soporte de aplicación (scaffolding)
- Generación de código
- Arquitectura Modelo Vista Controlador (MVC)
- Despachador de peticiones (dispatcher), con URLs y rutas personalizadas y limpias
- Validación integrada
- Plantillas rápidas y flexibles (sintaxis de PHP, con ayudantes “helpers”)
- Ayudantes para AJAX, Javascript, formularios HTML y más
- Componentes de Email, Cookie, Seguridad, Sesión y Manejo de solicitudes
- Listas de control de acceso flexibles
- Limpieza de datos
- Caché flexible
- Localización
- Funciona en cualquier subdirectorio del sitio web, con poca o ninguna configuración de Apache

Convenciones de configuración

CakePHP proporciona una estructura organizativa básica que cubre los nombres de las clases, archivos, tablas de base de datos y otras convenciones más. Aunque lleva algo de tiempo aprender las convenciones, siguiéndolas CakePHP evita que tener que hacer configuraciones innecesarias y hace que la estructura de la aplicación sea uniforme y que el trabajo con varios proyectos sea sencillo.

La capa modelo

La capa Modelo representa la parte de la aplicación que implementa la lógica de negocio. Es la responsable de obtener datos y convertirlos en los conceptos que utiliza la aplicación. Esto incluye procesar, validar, asociar u otras tareas relacionadas con el manejo de datos.

La capa vista

La capa Vista renderiza una presentación de datos modelados. Separada de los objetos Modelo, es la responsable de usar la información disponible para producir cualquier interfaz de presentación que pueda necesitar la aplicación.

Por ejemplo, la vista podría usar datos del modelo para renderizar una plantilla HTML que los contenga o un resultado en formato XML.

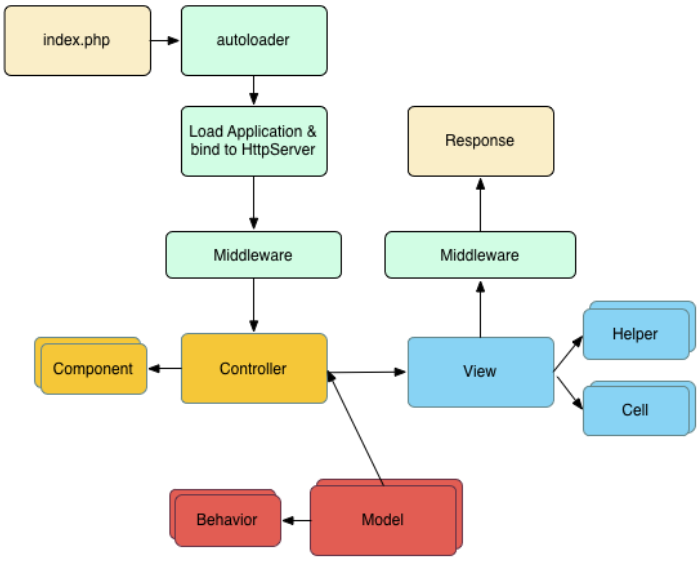
Esta capa no se limita a representaciones HTML o texto de los datos. Puede utilizarse para otros formatos habituales como JSON, XML y a través de una arquitectura modular, cualquier otro formato que se pueda necesitar

como CSV.

La capa controlador

La capa Controlador maneja peticiones de usuarios. Es la responsable de elaborar una respuesta con la ayuda de las capas Modelo y Vista. Un controlador puede verse como un gestor que asegura que todos los recursos necesarios para completar una tarea son delegados. Espera por las peticiones de los clientes, comprueba la validez de acuerdo con las reglas de autenticación y autorización, delega la búsqueda o procesado de datos al modelo, selecciona el tipo de presentación que el cliente acepta y finalmente delega el proceso de renderizado a la capa Vista.

Figura 7: Ciclo de un consulta en CakePHP



Fuente: http://cakephp.org/

El ciclo de petición típico de CakePHP comienza con un usuario solicitando una página o recurso en la aplicación. A un alto nivel cada petición sigue los siguientes pasos:

1. Las reglas de rescritura del servidor web envían la petición a webroot/index.php.
2. La aplicación es cargada y ligada a un HttpServer.
3. Se inicializa el midleware de la aplicación.
4. Una petición y respuesta son precesadas a través del Middleware PSR-7 que la aplicación utiliza. Normalmente esto incluye la captura de errores y enrutamiento.
5. Si no recibe ninguna respuesta del middleware y la petición contiene información de enrutamiento, se selecciona un controlador y una acción.
6. La acción del controlador es ejecutada y el controlador interactúa con los Modelos y Componentes necesarios.
7. El controlador delega la creación de la respuesta a la Vista para generar la salida a partir de los datos del modelo.
8. La vista utiliza Helpers y Cells para generar el cuerpo y las cabeceras de la respuesta.
9. La respuesta es devuelta a través del Middleware.
10. El HttpServer envía la respuesta al servidor web.

(Gonzales, 2014)

2.3.7. Normalización de Bases de datos

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener. También se puede entender la normalización como una serie de reglas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema que

minimice los problemas de lógica. Cada regla está basada en la que le antecede. La normalización se adoptó porque el viejo estilo de poner todos los datos en un solo lugar, como un archivo o una tabla de la base de datos, era ineficiente y conducía a errores de lógica cuando se trataban de manipular los datos.

La normalización también hace las cosas fáciles de entender. Los seres humanos tenemos la tendencia de simplificar las cosas al máximo. Lo hacemos con casi todo, desde los animales hasta con los automóviles. Vemos una imagen de gran tamaño y la hacemos más simple agrupando cosas similares juntas. Las guías que la normalización provee crean el marco de referencia para simplificar una estructura de datos compleja.

Otra ventaja de la normalización de base de datos es el consumo de espacio. Una base de datos normalizada ocupa menos espacio en disco que una no normalizada. Hay menos repetición de datos, lo que tiene como consecuencia un mucho menor uso de espacio en disco.

El proceso de normalización tiene un nombre y una serie de reglas para cada fase. Esto puede parecer un poco confuso al principio, pero poco a poco se va entendiendo el proceso, así como las razones para hacerlo de esta manera.

Grados de normalización

Existen básicamente tres niveles de normalización: Primera Forma Normal (1NF), Segunda Forma Normal (2NF) y Tercera Forma Normal (3NF). Cada una de estas formas tiene sus propias reglas.

Cuando una base de datos se conforma a un nivel, se considera normalizada a esa forma de normalización. No siempre es una buena idea tener una base de datos conformada en el nivel más alto de normalización, puede llevar a un nivel de complejidad que pudiera ser evitado si estuviera en un nivel más bajo de normalización.

- **Primera forma normal:** La regla de la Primera Forma Normal establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas. Poner la base de datos en la Primera Forma Normal resuelve el problema de los encabezados de columna múltiples. Muy a menudo, los diseñadores de bases de datos inexpertos harán algo similar a la tabla no normalizada. Una y otra vez, crearán columnas que representen los mismos datos. La normalización ayuda a clarificar la base de datos y a organizarla en partes más pequeñas y más fáciles de entender. En lugar de tener que entender una tabla gigantesca y monolítica que tiene muchos diferentes aspectos, sólo tenemos que entender los objetos pequeños y más tangibles, así como las relaciones que guardan con otros objetos también pequeños.
- **Segunda forma normal:** La regla de la Segunda Forma Normal establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas. Una dependencia parcial es un término que describe a aquellos datos que no dependen de la llave primaria de la tabla para identificarlos. Una vez alcanzado el nivel de la Segunda Forma Normal, se controlan la mayoría de los problemas de lógica. Podemos insertar un registro sin un exceso de datos en la mayoría de las tablas.

- **Tercer forma normal:** Una tabla está normalizada en esta forma si todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas. Comentamos anteriormente que una dependencia transitiva es aquella en la cual existen columnas que no son llave que dependen de otras columnas que tampoco son llave. Cuando las tablas están en la Tercera Forma Normal se previenen errores de lógica cuando se insertan o borran registros. Cada columna en una tabla está identificada de manera única por la llave primaria, y no deben haber datos repetidos. Esto provee un esquema limpio y elegante, que es fácil de trabajar y expandir. Un dato sin normalizar no cumple con ninguna regla de normalización. Para explicar con un ejemplo en que consiste cada una de las reglas, vamos a considerar los datos de la siguiente tabla.

La normalización es una técnica que se utiliza para crear relaciones lógicas apropiadas entre tablas de una base de datos. Ayuda a prevenir errores lógicos en la manipulación de datos. La normalización facilita también agregar nuevas columnas sin romper el esquema actual ni las relaciones.

Existen varios niveles de normalización: Primera Forma Normal, Segunda Forma Normal, Tercera Forma Normal, Forma Normal Boyce-Codd, Cuarta Forma Normal, Quinta Forma Normal o Forma Normal de Proyección-Unión, Forma Normal de Proyección-Unión Fuerte, Forma Normal de Proyección-Unión Extra Fuerte y Forma Normal de Clave de Dominio. Cada nuevo nivel o forma nos acerca más a hacer una base de datos verdaderamente relacional.

Se discutieron las primeras tres formas. Éstas proveen suficiente nivel de normalización para cumplir con las necesidades de la mayoría de las bases de datos. Normalizar demasiado puede conducir a tener una base de datos ineficiente y hacer a su esquema demasiado complejo para trabajar. Un balance apropiado de sentido común y práctico puede ayudarnos a decidir cuándo normalizar.

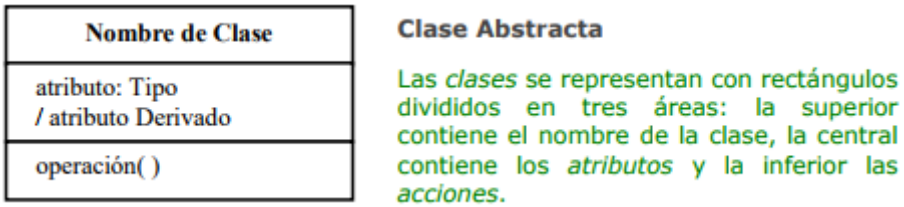
2.3.8. Diagramas UML

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Recordemos que un modelo es una representación simplificada de la realidad; el modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema. A continuación se describirán los diagramas más comunes del UML y los conceptos que representan:

- **Diagrama de clases:** Los diagramas de clases describen la estructura estática de un sistema. Las cosas que existen y que nos rodean se agrupan naturalmente en categorías. Una clase es una categoría o grupo de cosas que tienen atributos (propiedades) y acciones similares. Un ejemplo puede ser la clase “Aviones” que tiene atributos como el “modelo de avión”, “la cantidad de motores”, “la velocidad de crucero” y “la capacidad de carga útil”. Entre las acciones de las cosas de esta clase se encuentran: “acelerar”, “elevarse”, “girar”, “descender”, “desacelerar”. Un rectángulo es

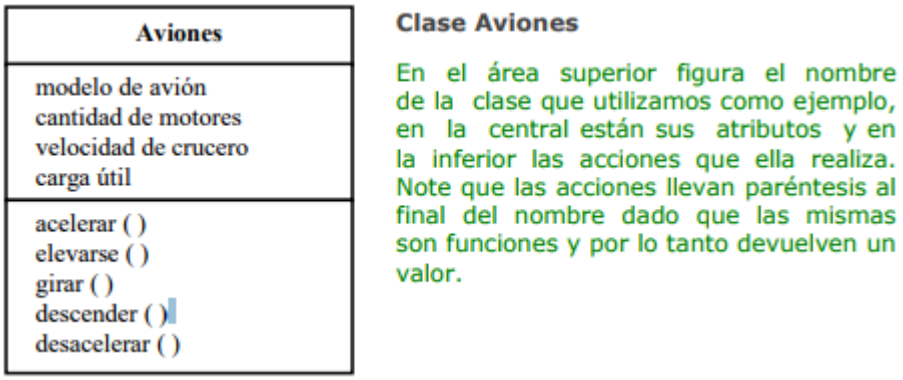
el símbolo que representa a la clase, y se divide en tres áreas. Un diagrama de clases está formado por varios rectángulos de este tipo conectados por líneas que representan las asociaciones o maneras en que las clases se relacionan entre sí.

Figura 8: Clase Abstracta



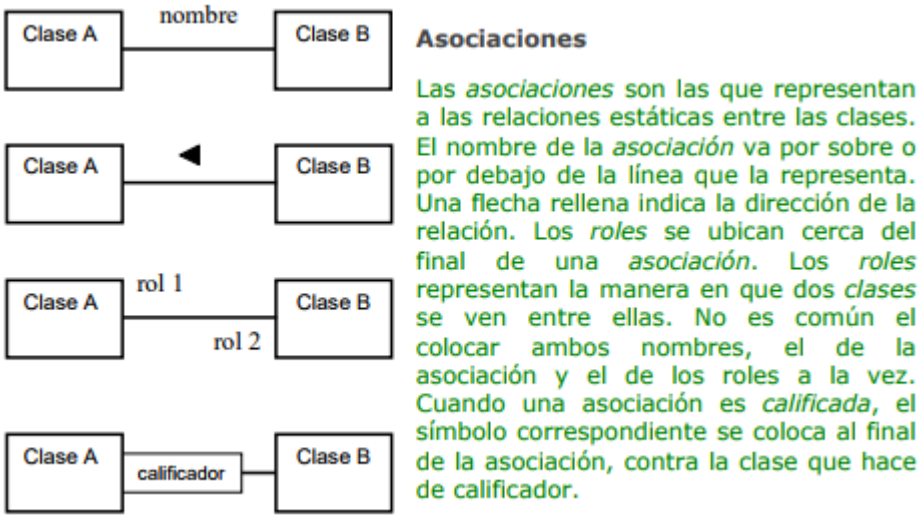
Fuente: Diagramas del UML, Hernández (2014)

Figura 9: Diagrama de clases, ejemplo de clases



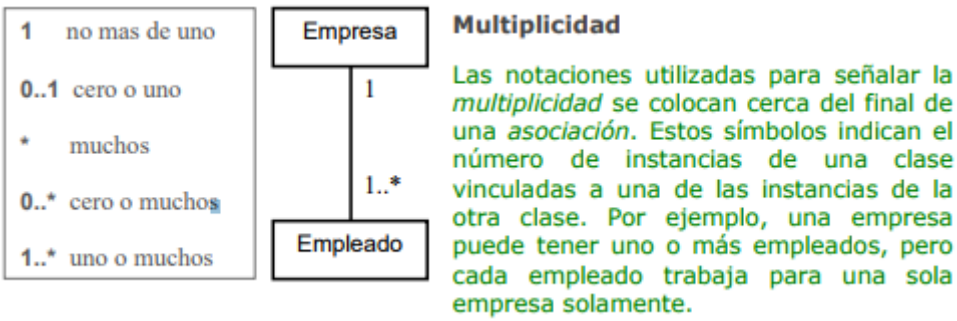
Fuente: Diagramas del UML, Hernández (2014)

Figura 10: Diagrama de clases, asociaciones



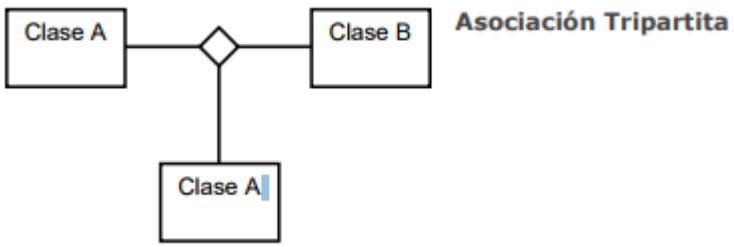
Fuente: Diagramas del UML, Hernández (2014)

Figura 11: Diagrama de clases – clase multiplicidad



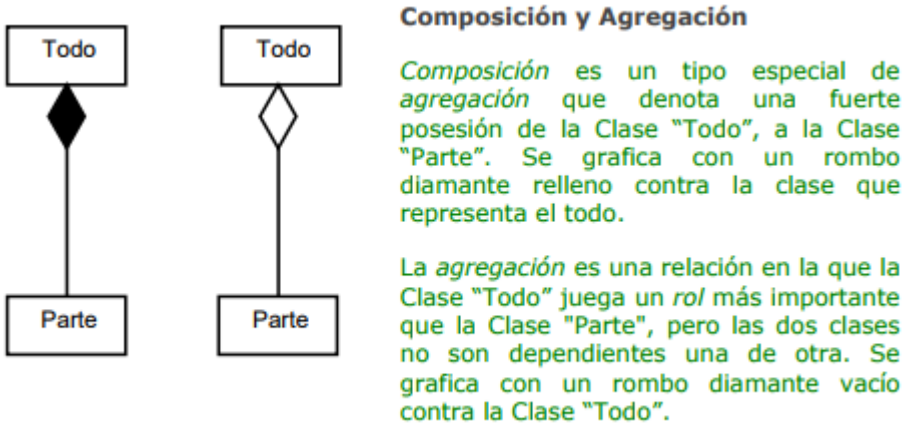
Fuente: Diagramas del UML, Hernández (2014)

Figura 12: Diagrama de clases – asociación tripartita



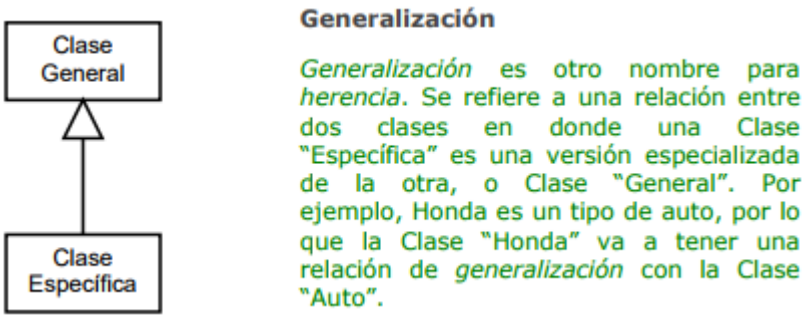
Fuente: Diagramas del UML, Hernández (2014)

Figura 13: Diagrama de clases – composición y agregación



Fuente: Diagramas del UML, Hernández (2014)

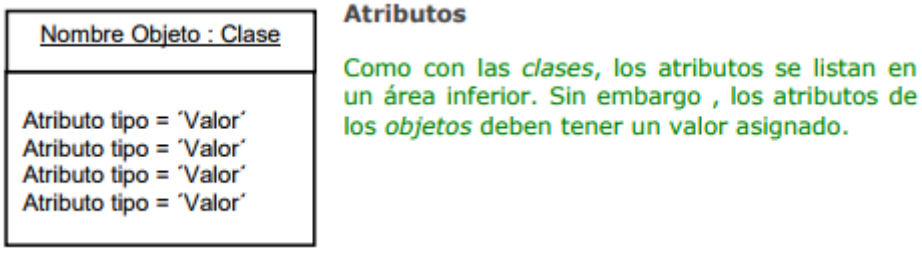
Figura 14: Diagrama de clases – generalización



Fuente: Diagramas del UML, Hernández (2014)

- **Diagrama de objetos:** Los Diagramas de Objetos están vinculados con los Diagramas de Clases. Un objeto es una instancia de una clase, por lo que un diagrama de objetos puede ser visto como una instancia de un diagrama de clases. Los diagramas de objetos describen la estructura estática de un sistema en un momento particular y son usados para probar la precisión de los diagramas de clases.

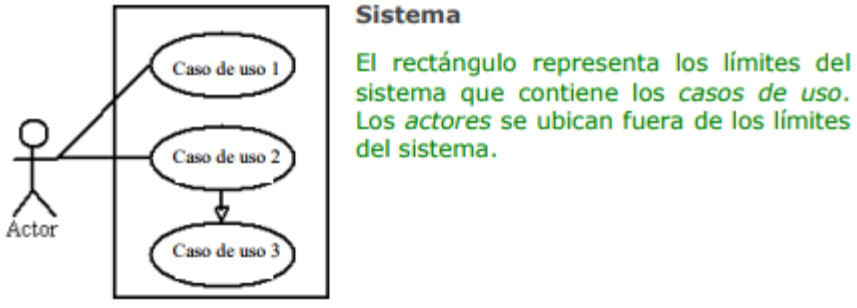
Figura 15: Figura de Objetos



Fuente: Diagramas del UML, Hernández (2014)

- **Diagrama de casos de uso:** Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Es una herramienta valiosa dado que es una técnica de aciertos y errores para obtener los requerimientos del sistema, justamente desde el punto de vista del usuario. Los diagramas de caso de uso modelan la funcionalidad del sistema usando actores y casos de uso. Los casos de uso son servicios o funciones provistas por el sistema para sus usuarios.

Figura 16: Diagrama de casos de uso



Fuente: Diagramas del UML, Hernández (2014)

2.3. DEFINICIÓN DE TÉRMINOS BÁSICOS

2.3.1. Sistema: Un sistema es un objeto complejo cuyos componentes se relacionan con al menos algún otro componente; puede ser material o conceptual. Todos los sistemas tienen composición, estructura y entorno, pero sólo los sistemas materiales tienen mecanismo, y sólo algunos sistemas materiales tienen figura o forma. (Bunge, 2001)

2.3.2. Información: La información es un conjunto organizado de datos procesados, que constituyen un mensaje que cambia el estado de conocimiento del sujeto o sistema que recibe el mensaje. (DLE, 2015)

Administración. Es el proceso cuyo objeto es la coordinación eficaz y eficiente de los recursos de un grupo social para lograr sus objetivos con la máxima productividad. (Münch, 2013)

2.3.3. Web: En informática, la World Wide Web (WWW) o red informática mundial es un sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía Internet. Con un navegador web, un usuario visualiza sitios web compuestos de páginas web que pueden contener texto, imágenes, vídeos u otros contenidos multimedia, y navega a través de esas páginas usando hiperenlaces. (Gonzálvez, 2011).

2.3.4. Intranet: Es una red informática que utiliza la tecnología del Protocolo de Internet para compartir información, sistemas operativos o servicios de computación dentro de una organización. Este término se utiliza en contraste con Extranet, una red entre las organizaciones, y en su lugar se refiere a una red dentro de una organización. A veces, el término se refiere únicamente a

la organización interna del sitio web, pero puede ser una parte más extensa de la infraestructura de tecnología de la información de la organización, y puede estar compuesta de varias redes de área local. El objetivo es organizar el escritorio de cada individuo con mínimo costo, tiempo y esfuerzo para ser más productivo, rentable, oportuno, seguro y competitivo. (Telleen,1998)

3.3.5. Framework: En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. (Riehle, 2000).

2.3.6. Front-end & Back-end: En diseño de software el Front-end es la parte del software que interactúa con el o los usuarios y el Back-end es la parte que procesa la entrada desde el Front-end. La separación del sistema en Front-end y Back-end es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas.

La idea general es que el Front-end sea el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas, y los transforma ajustándolos a las especificaciones que demanda el back-end para poder procesarlos, devolviendo generalmente una respuesta que el Front-end recibe y expone al usuario de una forma entendible para este. (Anónimo, 2015).

3.2.7. Usuario: Un usuario es aquel que usa algo o que usa ordinariamente algo (DLE, 2014) Por ejemplo un usuario de una biblioteca es un lector interesado en leer algún volumen de su archivo. Sin embargo, esto se opone a los conceptos de la Web semántica, Web 2.0 y 3.0, ya que la realidad actual prima a los ciudadanos como emisores y no solo como receptores que «usan» los medios. Es preferible, por tanto hablar de actores, sujetos, ciudadanos, etc. para referirse a las personas que interactúan en las redes digitales.

2.3.8. Docente: Un profesor, docente o enseñante, es quien se dedica profesionalmente a la enseñanza, bien con carácter general, bien especializado en una determinada área de conocimiento, asignatura, disciplina académica, ciencia o arte. (DLE, 2015)

2.3.9. Estudiante: La palabra estudiante es un sustantivo masculino o femenino que se refiere al educando o alumno o alumna dentro del ámbito académico, y que se dedica a esta actividad como su ocupación principal. (DLE, 2014)

2.3.10. CEPREUNA: El Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano es una dependencia cuyo fin es la formación y preparación de estudiantes del 4°, 5° grado y público en general para rendir el examen de admisión por esta modalidad.

CAPITULO III

MATERIALES Y MÉTODOS

3.1. POBLACION

La población tomada en cuenta para el desarrollo del presente estuvo conformada por los trabajadores del a cargo del desarrollo de los ciclos formativos del CEPREUNA comprendidos durante el ciclo formativo 215 – II.

Tabla 1: Población CEPREUNA ciclo 2015 – II según cargo.

Tipo	Cantidad
Comisión	3
Coordinadores	2
Auxiliares de educación	8
Auxiliares de sistemas	2
Total	**15**

Fuente: Elaboración Propia

3.2. MUESTRA

Por la naturaleza de la investigación para el presente, el cálculo de muestra no es necesario ya que la población total a la que se aplica el presente trabajo no supera las 20 unidades.

3.3. MÉTODO DE RECOLECCION DE DATOS

El método utilizado para la recolección de los datos ha sido la aplicación del cuestionario descrito en el ANEXO III basado en las métricas establecidas en la norma ISO 9126.

Para el análisis estadístico, se recurre a la aplicación de la diferencias de medias con la data obtenida aplicando los cuestionarios de los ANEXOS I y II que nos permiten comparar los tiempo de las tareas más recurrentes durante cada ciclo formativo.

Para el análisis y validación del software, se ha recurrido también a entrevistas personales con los miembros de la comisión y trabajadores del CEPREUNA.

3.4. SISTEMA DE VARIABLES

Para esta investigación se ha definido las siguientes variables.

Variable Independiente: Sistema para la Administración Académica del Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano Puno - 2015.

Variable dependiente: Mejora significativa de la Administración Académica del Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano Puno – 2015.

Tabla 2: Operacionalización de variables.

	DIMENSIÓN	INDICADOR	ESCALA	INSTRUMENTO
INDEPENDIENTE	SISTEMA PARA LA ADMINISTRACIÓN ACADÉMICA DEL CENTRO PREUNIVERSITARIO DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO.	Fiabilidad	5 - Muy confiable 4 - Confiable 3 - Regular 2 - Poco confiable 1 - Nada Confiable	Encuesta
		Portabilidad	5 - Muy fácil 4 - Fácil 3 - Regular 2 - Difícil 1 - Muy difícil	
		Mantenibilidad	5 - Muy fácil 4 - Fácil 3 - Regular 2 - Difícil 1 - Muy difícil	
DEPENDIENTE	MEJORA SIGNIFICATIVA DE LA ADMINISTRACIÓN ACADÉMICA DEL CENTRO DE ESTUDIOS PREUNIVERSITARIOS DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO PUNO - 2015	Eficiencia	5 – Muy eficiente 4 - Eficiente 3 - Regular 2 - Deficiente 1 - Muy deficiente	
		Usabilidad	5 - Muy fácil 4 - Fácil 3 - Regular 2 - Difícil 1 - Muy difícil	
		Funcionalidad	5 - Todas y más 4 - Todas 3 - Regular 2 - Muy pocas 1 - Ninguna	

Fuente: Elaboración Propia

3.5. PRUEBA DE HIPÓTESIS

Para la prueba de hipótesis, se recurre al método de diferencia de medias con un valor de $\alpha = 0.05$, comparando los tiempos que los usuarios tardan en realizar las tareas más recurrentes, para tal efecto se hacen uso de los cuestionarios adjuntos en los ANEXOS I y II.

3.6. METODOLOGIA DE DESARROLLO DE SOFTWARE (eXtreme Programming)

El desarrollo del sistema para la administración académica del CEPREUNA, se basa en la metodología de desarrollo XP (eXtreme Programming) por su dinamismo, simple implementación, fácil adecuación y la importancia que le da a la opinión del usuario y así mismo lo incluye en el equipo de desarrollo.

La ventaja de XP frente a otras metodologías, es su orientación hacia los resultados, mientras que en otras las actividades a realizar son tantas que el ritmo entero del desarrollo se retarda y el factor tiempo es de importancia en la presente investigación.

La tarea desarrollo e implementación ha sido dividida en 4 fases según XP, para separar y organizar las acciones a desarrollar con el fin de optimizar todo el proceso de implementación.

3.6.1. Fase de exploración

XP propone en esta fase determinar los alcances del proyecto y definir las necesidades del mismo haciendo uso de las historias de usuario, para esta

investigación el proceso de recolección de historias de usuario ha sido reemplazado por la participación y observación in situ de todo el proceso que conlleva el desarrollo de un ciclo formativo, salvo algunas excepciones se han realizado entrevistas directas con las comisión esto para aclarar la aplicación de algunas reglas especificadas en el reglamento de admisión CEPREUNA 2015.

3.6.2. Fase de Planificación

Para esta fase, se establecen reuniones con los usuarios, miembros de la comisión, esto con el fin de determinar el orden en la que debe implementar todas las funcionalidades requeridas según la Tabla 3 y establecer las fechas del plan de entregas tomando en cuenta el orden de cada uno de los procesos durante el ciclo.

3.6.3. Fase de Iteración

Para el desarrollo de esta fase, se ha contado con el apoyo de los usuarios como parte activa del equipo de desarrollo, como el resultado de la fase de exploración no tiene un suficiente detalle que permita su análisis, se recurre al apoyo del o los usuarios con el fin de verificar que cada entregable cumpla con las tareas necesarias tras la codificación y desarrollo.

3.6.4. Fase de puesta en producción

Durante esta fase y con la aprobación del usuario para este caso los miembros de la comisión, se pone en operación los módulos funcionales y se hacen los ajustes necesarios ya que muchas veces el entorno de pruebas no suele reflejar con exactitud el entorno en el que verdaderamente de ejecutarán

los módulos.

3.7. REQUERIMIENTOS DEL SISTEMA

Requerimientos funcionales

De la Tabla 2 y de las reuniones sostenidas con el personal directamente implicado en el desarrollo de cada ciclo formativo, se han tomado como primordiales el desarrollo de los siguientes módulos o componentes:

Tabla 3: Módulos y acciones a implementar

N°	Módulo	Acciones	Prioridad
1	Administración	• Registrar estudiantes. • Registrar docentes. • Validar matrículas. • Validar registro de docentes. • Asignar salones. • Validar pagos. • Registrar asistencia para docentes y estudiantes. • Emitir constancias de no adeudo. • Emitir reportes consolidados. • Administrar usuarios y perfiles de acceso.	Alta
2	Estudiante	• Registro de estudiante. • Registro de pagos. • Consulta de asistencias. • Descarga de cuadernillos.	Alta
2	Docente	• Registro de docentes- • Consulta de asistencias. • Publicación de cuadernillos de trabajo. • Descarga de cuadernillos de trabajo	Alta

Fuente: Elaboración Propia

Requerimientos no funcionales

- Interfaz amigable y adaptable (responsive).
- Disponibilidad.
- Estabilidad.
- Portabilidad.
- Seguridad.

Requerimientos técnicos

Tabla 4: Requerimientos técnicos del sistema

Tipo	Herramienta/Servicio
Software	• PHP 5.6 • CakePHP 3.15 • Mysql 5 / MariaDB 10 • Atom Text Editor • Mysql WorkBench 6 • Apache 2.4 / Nginx 1.12 • Adobe Flash Player • Jquery 1.8 o superior • Bootstrap CSS 3.6 • Chrome 50 / Firefox 3.6
Hardware	• PC core i3 o superior • Impresora • Lector de código de barras
Servicios	• Conexión a internet 4G / ADSL / HFC

Fuente: Elaboración Propia

CAPITULO IV

RESULTADOS Y CONCLUSIÓN

4.1. ANÁLISIS DEL SISTEMA

SITUACION INICIAL

El CEPREUNA, ha tenido implementado, dentro de su forma de administración, prototipos independientes que de alguna forma socapan el arduo trabajo que implica el manejo de gran cantidad de información, lo que ha venido generado demoras, errores, y la poca integridad que ofrecían los datos registrados. En varios de los casos la utilización de estos prototipos ha causado más demoras a comparación del manejo manual.

DEFINICION DE ROLES

Dada la coyuntura de la investigación, los disponibilidad de recursos humamos, el investigador ha asumido los roles de directa relación con el desarrollo del sistema. Solo se han tomado en consideración los roles más importantes según el desarrollo del presente.

- **Programador:** El investigador asume el rol de programador por tal motivo es el encargado de escribir el código fuente necesario para la implementación del sistema.
- **Cliente:** El CEPREUNA cumple el rol de cliente, define las especificaciones del sistema e influye en el desarrollo sin ejercer control, define las pruebas funcionales.
- **Tester:** Este rol es también asumido por el desarrollador con el fin de apoyar al cliente en la preparación y realización de prueba funcional y mediante esta explicar los resultados al equipo.
- **Tracker:** El investigador analiza la información sobre la marcha del proyecto sin afectar demasiado el proceso.
- **Entrenador:** El investigador, es el responsable global del proyecto también es el encargado de verificar que se estén aplicando correctamente las guías XP.

4.2. PLANIFICACION

El sistema ha sido desarrollado siguiendo los pasos establecidos en la metodología XP, si bien es cierto se han reemplazado u obviado alguno de sus procedimientos, ya que la aplicación de estos no ha sido necesaria o se ha hecho uso de otro método con el fin de obtener el mismo resultado en el desarrollo del sistema.

Es necesario mencionar también que el desarrollo ha estado sujeto a variaciones y cambios intempestivos, se han aplicado también procesos de optimización de código fuente para garantizar un alto rendimiento.

4.2.1. Fase de exploración

De la observación y la participación durante los procesos que de desarrollan durante el desarrollo de cada ciclo formativo, se ha podido establecer un tiempo estimado de desarrollo que obviamente está sujeto a variaciones. La participación y observación in situ de los procesos que se desarrollan durante el ciclo formativo arrojan al equipo de desarrollo un resultado con más detalle para su análisis, en tal efecto podemos decir que la estimación de tiempos de entrega puede hacerse con un nivel más elevado de certeza.

Tabla 5: Tiempos estimados de desarrollo

N°	Módulo	Acciones	T. Estimado
1	Administración	• Registrar estudiantes. • Registrar docentes. • Validar matrículas. • Validar registro de docentes. • Asignar salones. • Registrar pagos. • Validar pagos. • Registrar asistencia para docentes y estudiantes. • Emitir constancias de no adeudo. • Emitir reportes consolidados. • Administrar usuarios y perfiles de acceso.	3 Semanas
2	Estudiante	• Registro de estudiante. • Registro de pagos. • Consulta de asistencias. • Descarga de cuadernillos.	2 Semanas
2	Docente	• Registro de docentes- • Consulta de asistencias. • Publicación de cuadernillos de trabajo. • Descarga de cuadernillos de trabajo	2 Semanas
TIEMPO TOTAL ESTIMADO DE DESARROLLO			**7 Semanas**

Fuente: Elaboración Propia

4.2.2. Fase de planificación

Después de haberse aprobado los tiempos estimados de desarrollo, lo que en esta fase se hizo es ordenar las prioridades y agrupar según las necesidades las historias de usuario. El cronograma de entregas o *release plan* se realizó en base a las estimaciones de tiempos de desarrollo realizada por el equipo.

XP propone la realización de reuniones luego de algunas iteración con el fin de darle una nueva revisión al plan de entregas para determinar si este debe ser ajustado o no.

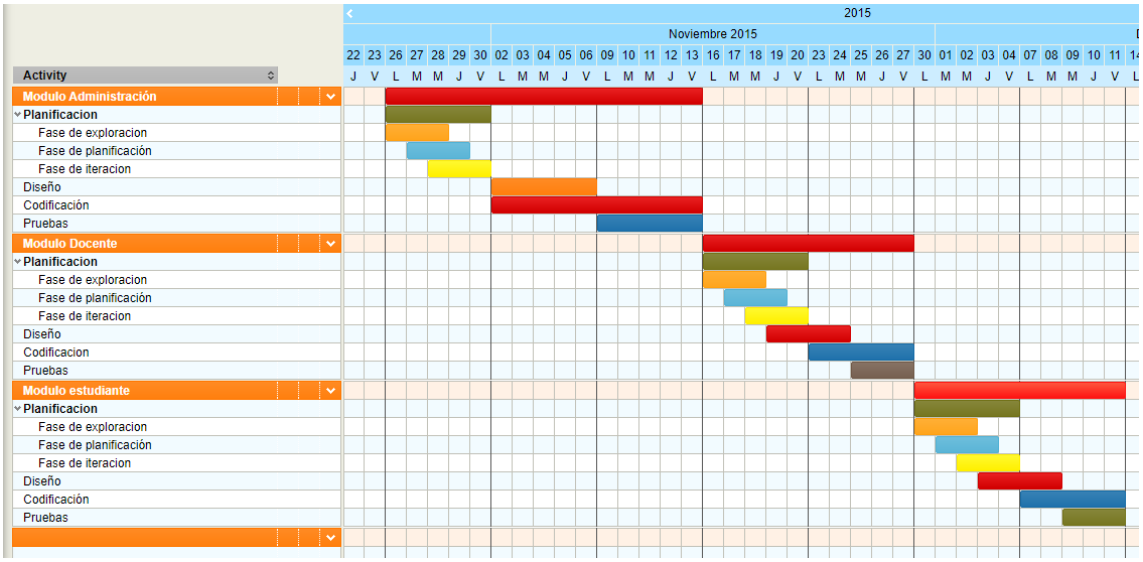
Tabla 6: Orden de prioridades de desarrollo

N°	Módulo	Acciones (En orden de prioridad)	Prioridad
1	Administración	1. Registrar estudiantes. 2. Registrar docentes. 3. Validar matrículas. 4. Validar registro de docentes. 5. Asignar salones. 6. Registrar pagos. 7. Validar pagos. 8. Registrar asistencia para docentes y estudiantes. 9. Emitir constancias de no adeudo. 10. Emitir reportes consolidados. 11. Administrar usuarios y perfiles de acceso.	1
2	Estudiante	12. Registro de estudiante. 13. Registro de pagos. 14. Consulta de asistencias. 15. Descarga de cuadernillos.	3
3	Docente	16. Registro de docentes- 17. Consulta de asistencias. 18. Publicación de cuadernillos de trabajo. 19. Descarga de cuadernillos de trabajo	2

Fuente: Elaboración Propia

Tomando en cuenta el orden de prioridades para el desarrollo y los tiempos estimados de desarrollo, se elabora el plan de entregas:

Figura 17: Tiempo estimado de desarrollo



Fuente: Elaboración Propia

4.2.3. Fase de iteraciones

Para este caso, todas las acciones determinadas durante la fase de exploración han sido convertidas en tareas específicas de programación, de acuerdo al orden preestablecido.

Asimismo, para cada acción de usuario se establecen las pruebas de aceptación, estas pruebas se realizaron al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no hayan afectado las anteriores.

4.2.4. Fase de puesta en producción

En este tramo de la investigación, una vez concluida con la fase de pruebas y con la aprobación del usuario, los módulos desarrollados son transferidos desde el ambiente de desarrollo hacia el sistema en funcionamiento, se realizan también los ajustes necesarios para que la integración del nuevo módulo no haga afecte el funcionamiento del sistema.

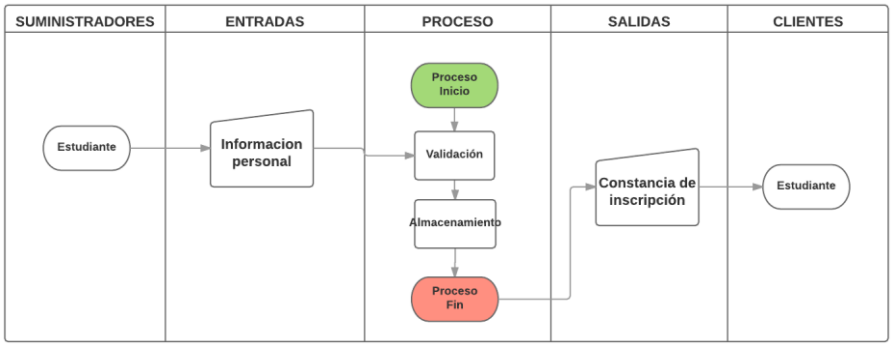
4.3. DISEÑO

Se ha **tomado** en consideración el especial énfasis que hace XP en los diseños simples y claros. Para un mejor entendimiento de las propiedades del sistema y aplicando los diagramas UML se presentan los siguientes gráficos.

Figura 18: Diagramas de casos de uso

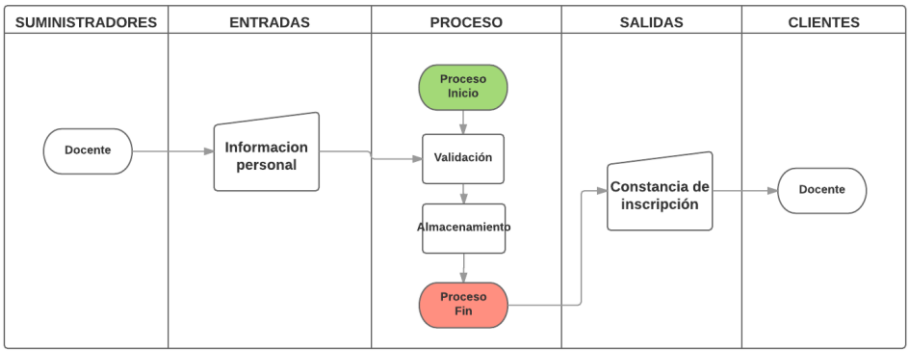
Fuente: Elaboración Propia

Figura 19: Proceso de registro de estudiantes



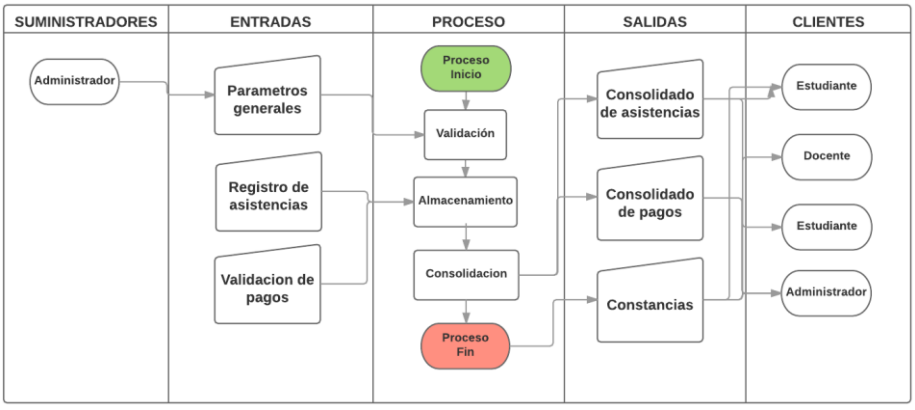
Fuente: Elaboración Propia

Figura 20: Proceso de registro de docentes.



Fuente: Elaboración Propia

Figura 21: Proceso del módulo de administración



Fuente: Elaboración Propia

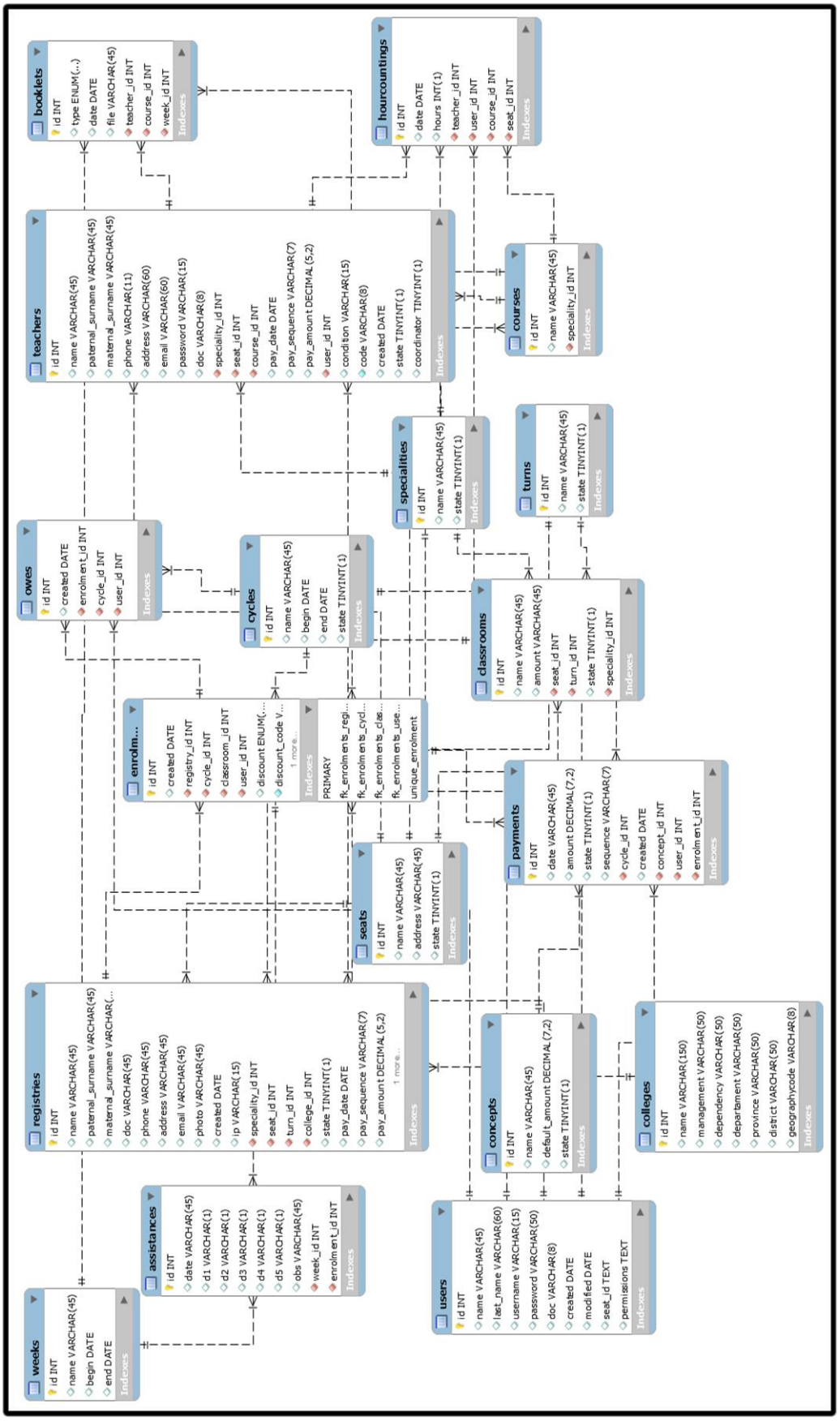


Figura 22: Diagrama E-R para modelado de la base de datos

Fuente: Elaboración Propia.

4.4. CODIFICACIÓN

La codificación del software se realiza según lo recomendado por XP, haciendo uso del lenguaje del lado del servidor PHP a través del framework CakePHP que a su vez es compatible con bibliotecas basadas en JavaScript tales como JQuery que facilitan el uso de Ajax y otros beneficios.

La utilización de CakePHP, nos obliga a adoptar cierto tipo de estándares en su codificación para este caso, se hacen uso de los estándares de codificación PSR7, de tal forma que así se reduce la complejidad de los trabajos de mantenimiento y escalado del sistema cumpliéndose así lo recomendado por XP.

Figura 23: Estándar de codificación para controladores

```php
<?php
    class EstudianteController extends AppController{
        public uses=array();        //declaramos las tablas necesarias
        public helpers=array();     //Declaramos lo helpers necesarios

        public function index(){
            //accion lógica para la vista index
        }

        public function registro(){
            //accion lógica para la vista registro
        }

    }
?>
```

Fuente: Elaboración Propia.

Figura 24: Estándar de codificación para modelos

```php
<?php
    class Estudiante extends AppModel{
        public $belongsTo=array();  //invoca las tablas relacionadas de 1:n
        public $hasMany=array();    //invoca las tablas relacionadas de m:1
        public $validate();         //establece las reglas de validacion de
                                    //datos

        public function Consulta(){
            //estructura para la consulta a la base de datos
        }

    }
?>
```

Fuente: Elaboración Propia.

Figura 25: Estándar de codificación para Helpers

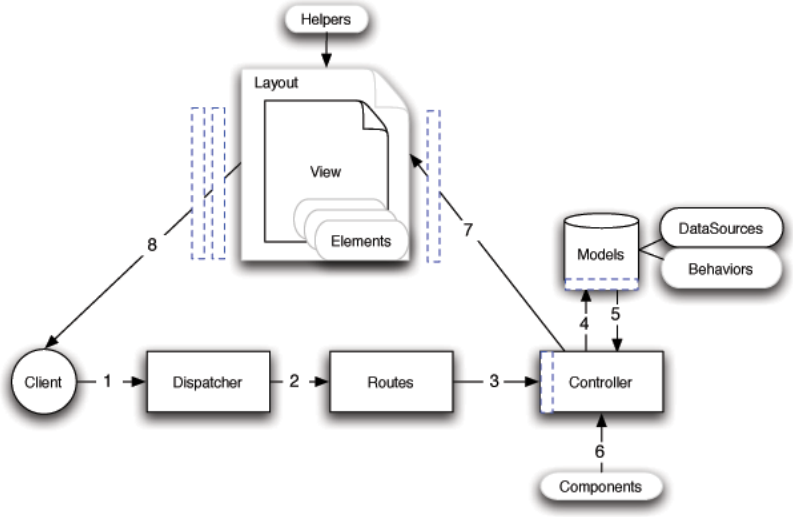
```php
<?php
    class AccionHelper extends AppHelper{
        public $helper=array();     // declaramos otros helpers si los
                                    // necesitamos

        public function contar(){
            // accion logica para la function contar
        }
    }
?>
```

Fuente: Elaboración Propia.

Para la codificación del sistema, se hace uso del patrón MVC que nativamente proporciona CakePHP.

Figura 26: Estructura del funcionamiento de CakePHP



Fuente: Elaboración Propia.

4.5. PRUEBAS

Si bien es cierto, XP recomienda la realización de pruebas al finalizar cada iteración, la coyuntura en la que se ha desarrolla el presente ha llevado a que el software este implementado con un log que permite la captura de eventos inusuales durante la ejecución de la prueba, ya que los cambio suelen realizarse con una solicitud mínima de tiempo, llegando a veces a realizarse pruebas en producción.

Figura 27: Log de captura de eventos inusuales o errores.

```
2017-09-19 17:46:17 Error: [Cake\Routing\Exception\MissingControllerException] Controller
class Cepre could not be found.
Exception Attributes: array (
  'class' => 'Cepre',
  'plugin' => false,
  'prefix' => false,
  '_ext' => false,
)
Request URL: /cepre/css/lib/bootstrap/bootstrap.css.map
Stack Trace:
#0 D:\Server\htdocs\tesis\vendor\cakephp\cakephp\src\Http\ControllerFactory.php(70):
Cake\Http\ControllerFactory->missingController(Object(Cake\Http\ServerRequest))
#1 D:\Server\htdocs\tesis\vendor\cakephp\cakephp\src\Http\ActionDispatcher.php(90):
Cake\Http\ControllerFactory->create(Object(Cake\Http\ServerRequest), Object(
Cake\Http\Response))
#2 D:\Server\htdocs\tesis\vendor\cakephp\cakephp\src\Http\BaseApplication.php(78):
Cake\Http\ActionDispatcher->dispatch(Object(Cake\Http\ServerRequest), Object(
Cake\Http\Response))
```

Fuente: Elaboración Propia.

4.6. INSTALACIÓN

El sistema, cuenta con un proceso de instalación sencillo, la simple colocación de la carpeta conteniendo los archivos dentro del directorio root de Apache sea local o remoto bastará para que el sistema entre en funcionamiento.

Para el pase a producción de alguna modificación realizada, bastara con reemplazar los archivos correspondientes, de hecho la modularidad con la que se ha desarrollado el sistema facilita la localización de estos.

4.7. RESULTADOS

Para el análisis de resultados (Anexo III), se ha aplicado la escala definida en el ANEXO II.

Tabla 7: Tabla de decisión ISO 9126

	INTERVALO	DECISIÓN
Inaceptable	[15 - 33>	
Mínimamente aceptable	[33 - 51>	
Aceptable	[51 - 69>	
Cumple los requisitos	[69 - 87>	77.53
Excede los requisitos	[87 - 105>	

Fuente: Elaboración Propia.

Conforme a los resultados obtenidos tras el procesamiento de los datos recolectados podemos afirmar que el SISTEMA PARA LA ADMINISTRACION ACADEMICA DEL CENTRO PREUNIVERSITARIO DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO – PUNO, cumple los requisitos con una puntuación promedio de 77.53.

4.8. ANÁLISIS ESTADÍSTICO

Para contrastar la hipótesis de la presente investigación se utiliza la prueba de diferencia de medias a través de la cual determinaremos si se acepta o rechaza la hipótesis planteada.

$$t_c = \frac{|\overline{X}_A - \overline{X}_B|}{\sqrt{\frac{\sigma_A^2}{n_A} - \frac{\sigma_B^2}{n_B}}}$$

Donde:

$\overline{X}_A$: Tiempo promedio de atención antes de la implementación del sistema.

$\overline{X}_B$: Tiempo promedio de atención después de la implementación del sistema.

S_A^2, S_B^2 : Varianza de A y B respectivamente.

n: Tamaño de la muestra

PRUEBA DE HIPÓTESIS

$H_0 : \mu_A = \mu_B$

La puesta en marcha del sistema administrativo no mejora la administración académica del Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano – Puno.

$H_1 : \mu_A < \mu_B$

La puesta en marcha de un sistema administrativo mejora significativamente la administración académica del Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano – Puno.

NIVEL DE SIGNIFICANCIA

El nivel de significancia aplicado es de $\propto = 0.05 = 5\%$ (equivalente a una confianza del 95%) y $n - 1$ grados de libertad donde: $T(0.05, n-1) =$

$T(0.05, 14) = 1.7613$

REGLA DE DECISIÓN

Si $t_c > t_t$ se rechaza la H_o y aceptamos H_1

PRUEBA ESTADÍSTICA

Para la prueba estadística se hizo uso de los resultados del cuestionario para medir los tiempos de las tareas más recurrentes antes de la implementación del sistema y el cuestionario para medir los tiempos de las tareas más recurrentes después de la implementación del sistema para mejorar la administración académica del Centro Preuniversitario de la Universidad Nacional del Altiplano. (ANEXOS I y II)

Tabla 8: Puntaje por usuario aplicando los cuestionarios de los ANEXOS I y II

USUARIO	ANTES	DESPUES
1	30	20
2	28	21
3	31	19
4	29	15
5	29	20
6	25	21
7	35	23
8	33	25
9	31	25
10	28	24
11	29	25
12	25	23
13	30	20
14	31	28
15	26	22
$\overline{X}$	**29.33**	**22.07**
σ^2	**7.67**	**10.07**

Fuente: Elaboración propia

Donde:

$n = 15$

$\overline{X}_A = 29.33 \rightarrow punjaje\ promedio\ antes\ de\ la\ implementación$

$\overline{X}_B = 22.07 \rightarrow Punaje\ promedio\ después\ de\ la\ implementación$

$\sigma_A^2 = 7.67$

$\sigma_B^2 = 10.07$

$$t_c = \frac{|\overline{X}_A - \overline{X}_B|}{\sqrt{\frac{\sigma_A^2}{n_A} + \frac{\sigma_B^2}{n_B}}} = \frac{|29.33 - 22.07|}{\sqrt{\frac{7.67}{15} + \frac{10.07}{15}}} = 6.6832188$$

DECISION

Como $t_c = 6.6832188 > t_t = 1.7613$, rechazamos H_0 y aceptamos H_1, pudiendo así afirmar que el la puesta en marcha de un sistema administrativo mejora significativamente la administración académica del Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano – Puno.

CONCLUSIONES

Se ha logrado desarrollar un sistema para la administración académica del Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano – Puno, mejorando significativamente la gestión de información académica, permitiendo realizar de manera mucho más rápida y optima todos los procesos en cada uno de sus ciclos formativos.

Se logró a través de la metodología XP analizar los requerimientos del Centro Preuniversitario de la Universidad Nacional del Altiplano – Puno, para determinar las funcionalidades y necesidades que debe satisfacer el sistema.

Con base en la metodología de desarrollo XP y utilizando CakePHP, se logró desarrollar un software portable, escalable, confiable y acorde a las necesidades del Centro Preuniversitario de la Universidad Nacional del Altiplano.

Se aplicaron las métricas de aceptación de software establecidas en la ISO 9126 quedando demostrado que el sistema cumple con satisfacer las necesidades de la institución habiendo obtenido una puntuación de 69.4 según la escala de calificación utilizada (ANEXO II), quedando así demostrado que se ha cumplido con desarrollar un software funcional, fiable, usable, eficiente, mantenible y portable.

RECOMENDACIONES

Se recomienda al CEPREUNA, renovar y mejorar sus procesos a través de la implementación de nuevas políticas que faciliten el desarrollo de cada ciclo formativo con el fin de mejorar los aspectos de manejo y gestión de información académica con el fin de que se pueda brindar un mejor servicio a los estudiantes.

XP ha demostrado ser una metodología capaz de organizar los diferentes procesos de desarrollo de software; por tanto recomendamos su utilización de forma más profunda con el fin de realizar mejores análisis y estimaciones de tiempos para asegurar el cumplimiento de los plazos establecidos.

Haciendo uso de las metodologías y herramientas descritas en la presente, sugerimos la un estudio más profundo de las bondades de los frameworks no solo del lado del servidor ya que CakePHP ha demostrado ser una herramienta de desarrollo rápido lo bastante robusta y confiable.

Si bien es cierto la aplicación de las métricas establecidas en la ISO 9126 ha servido para evaluar la aceptación del software, sugerimos aplicar nuevas formas de validación de software acorde a la realidad de cada entorno.

BIBLIOGRAFIA

Casallas, R. (2015). *Ingeniería del Software: Ciclos de vida y metodologías*. Recuperado de: ttp://sistemas.uniandes.edu.co/~isis2603/dokuwiki/lib/exe/fetch.php?media=principal:isis2603-modelosciclosdevida.pdf

Caceres, R., Colchado, W. (2014) IMPLEMENTACIÓN DE UN SISTEMA WEB PARA LOS PROCESOS DE ADMISIÓN DE LA USMP ORIENTADO A LA NORMA ISO 9001 (Tesis de pregrado). Universidad San Martin de Porres, Lima, Perú.

Chambilla, C. (2017) *IMPLEMENTACIÓN Y DESARROLLO DE SISTEMAS AUTOMATIZADOS PARA MEJORAR LA PROGRAMACIÓN Y EJECUCIÓN DE PROCESOS EN LA GESTIÓN MUNICIPAL, BASADOS EN ARQUITECTURA JAVA EE – SPRING (Tesis de pregrado)*. Universidad Nacional de San Agustín, Arequipa. Perú

Cockburn, A., Williams, L. (2000). *The Costs and Benefits of Pair Programming.* Recuperado de: ttp://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF

DICCIONARIO DE LA LENGUA ESPAÑOLA, (2014, 2015)

Domínguez, R. (2016), Aplicación de métricas de calidad en uso utilizando la ISO 9126 para determinar el grado de satisfacción del Sistema Único de Matrícula *(Tesis de pregrado)*. Universidad Nacional Mayor de San Marcos, Lima, Perú.

Fowler, M. (2015). *The New Methodoly.* Recuperado de: https://martinfowler.com/articles/newMethodology.html.

Gonzales J. (2014). *Rapid Application Development with CakePHP 2*.

Gonzálvez, V. (2011), *La Web 2,0 Y 3,0 en su relación con el EEES*, España: Visión Libros.

Romero, R. (2013), *ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN APLICADO A LA GESTIÓN EDUCATIVA EN CENTROS DE EDUCACIÓN ESPECIAL (Tesis de pregrado).* Pontificia Universidad Católica del Perú, Lima, Perú.

Tomayko, J, Herbsleb, J. (2003). *How Useful Is the Metaphor Component of Agile Methods? A Preliminary Study.* USA: School of Computer Science: Carnegie Mellon.

Shaikh, A., Karjaluoto, H. (2015) *Making the most of information technology & systems usage: A literature review, framework and future research agenda.* Finlandia.

Silberchatz, A. (2014), *Fundamentos de bases de datos*, Madrid: McGrawHill.

Venners, B. (2003). *Collective Ownership of Code and Text, A Conversation with Ward Cunningham.* USA: Artima Developer

Wells, D. (2013). *Extreme Programming: A gentle introduction.* Recuperado de http://www.extremeprogramming.org/

ANEXOS

ANEXO I. Cuestionario para medir los tiempos de las tareas más recurrentes antes de la implementación del sistema para mejorar la administración académica del Centro de Estudios Preuniversitarios de la Universidad Nacional del Altiplano.

N°	PREGUNTA	ESCALA				
		1	2	3	4	5
1	¿Cuánto tiempo demora el proceso de inscripción para estudiantes?					
2	¿Cuánto tiempo demora el proceso de inscripción para docentes?					
3	¿Cuál es el tiempo medio que lleva generar consultas y reportes?					
4	¿Cuál es el tiempo medio en el que se generan los carnés para estudiantes?					
5	¿Cuál es el tiempo en el que se generan las constancias de no adeudo para estudiantes?					
6	¿Cuál es el tiempo promedio que le lleva el registro y validación de pagos?					
7	¿Cuál es el tiempo promedio que le lleva el registro y control de asistencia diaria para estudiantes?					
8	¿Cuál es el tiempo medio que le lleva el registro, control y conteo de asistencia diaria de docentes?					
9	¿Cuál es el tiempo aproximado que le toma obtener la información consolidada?					

Fuente: Elaboración propia

LEYENDA	
1	0m – 30m
2	30m – 1h
3	1h – 1:30h
4	1:30h – 2h
5	2h – a más

Fuente: Elaboración propia

ANEXO II. Cuestionario para medir los tiempos de las tareas más recurrentes después de la implementación del sistema para mejorar la administración académica del Centro Preuniversitario de la Universidad Nacional del Altiplano.

N°	PREGUNTA	ESCALA				
		1	2	3	4	5
1	¿Cuánto tiempo demora el proceso de inscripción para estudiantes tras la implementación del sistema?					
2	¿Cuánto tiempo demora el proceso de inscripción para docentes tras la implementación del sistema?					
3	¿Cuál es el tiempo medio que lleva generar consultas y reportes tras la implementación del sistema?					
4	¿Cuál es el tiempo medio en el que se generan los carnés para estudiantes?					
5	¿Cuál es el tiempo en el que se generan las constancias de no adeudo para estudiantes tras la implementación del sistema?					
6	¿Cuál es el tiempo promedio que le lleva el registro y validación de pagos tras la implementación del sistema?					
7	¿Cuál es el tiempo promedio que le lleva el registro y control de asistencia diaria para estudiantes tras la implementación del sistema?					
8	¿Cuál es el tiempo medio que le lleva el registro, control y conteo de asistencia diaria de docentes tras la implementación del sistema?					
9	¿Cuál es el tiempo aproximado que le toma obtener la información consolidada tras la implementación del sistema?					

Fuente: Elaboración propia

LEYENDA	
1	0m – 30m
2	30m – 1h
3	1h – 1:30h
4	1:30h – 2h
5	2h – a más

Fuente: Elaboración propia

ANEXO III. Cuestionario de evaluación de la aceptación del software para la Administración académica del Centro Preuniversitario de la Universidad Nacional del Altiplano Puno.

INDICADORES	PREGUNTAS	PUNTUACIÓN				
		1	2	3	4	5
FUNCIONALIDAD						
Idoneidad	¿El software desempeña las tareas para las cuales fue diseñado?					
Exactitud	¿Los resultados que obtiene del software son consistentes y cumplen con lo esperado?					
Interoperabilidad	¿El sistema puede actuar con otros sistemas independientes sin afectar su funcionamiento?					
Seguridad	¿El sistema puede impedir el acceso a personal no autorizado?					
FIABILIDAD						
Madurez	¿El sistema funciona sin presentar fallas frecuentes?					
Recuperabilidad	¿El sistema funciona sin perder datos por fallas ocasionales?					
Tolerancia a fallos	¿Las fallas ocasionales del sistema no afectan gravemente el funcionamiento del sistema?					
USABILIDAD						
Aprendizaje	¿Qué tan fácil es aprender a usar el sistema?					
Comprensión	¿Qué tan fácil es comprender el funcionamiento del sistema?					
Operatividad	¿El sistema puede ser operado sin mucho esfuerzo?					
Atractividad	¿Qué tan atractiva es la apariencia del software?					
EFICIENCIA						
Comportamiento en el tiempo	¿Qué tan rápido responde el sistema?					
Comportamiento de recursos	¿El software, qué tan eficientemente maneja los recursos del sistema?					
MANTENIBILIDAD						
Estabilidad	¿Qué tan estable es el software a pesar de los cambios?					
Facilidad de análisis	¿Qué tan fácil es detectar las fallas en el software?					
Facilidad de cambio	¿Qué tan fácil es modificar el software?					
Facilidad de pruebas	¿Qué tan fácil es realizar pruebas en el sistema?					
PORTABILIDAD						
Capacidad de instalación	¿Qué tan fácil es instalar el software?					
Capacidad de reemplazamiento	¿Qué tan fácil es reemplazar el software por otro similar?					
Adaptabilidad	¿Qué tan fácil es trasladar el software a otros ambientes?					
Co-existencia	¿El software puede funcionar con otros sistemas?					

Fuente: Elaboración propia

ANEXO IV. Escala valorativa utilizado para la evaluación de aceptación del software**.**

	INTERVALO
Inaceptable	[15 - 33>
Mínimamente aceptable	[33 - 51>
Aceptable	[51 - 69>
Cumple con los requisitos	[69 - 87>
Excede requisitos	[87 - 105>

Fuente: Elaboración propia

ANEXO V. Conteo y evaluación de los resultados de la aplicación del cuestionario (ANEXO III)

INDICADORES	PREGUNTAS	CONTEO					RESULTADO				
		1	2	3	4	5	1	2	3	4	5
FUNCIONALIDAD											
Idoneidad	¿El software desempeña las tareas para las cuales fue diseñado?	0	0	2	9	4	0	0	6	36	20
Exactitud	¿Los resultados que obtiene del software son consistentes y cumplen con lo esperado?	0	0	5	8	2	0	0	15	32	10
Interoperabilidad	¿El sistema puede actuar con otros sistemas independientes sin afectar su funcionamiento?	0	0	5	8	2	0	0	15	32	10
Seguridad	¿El sistema puede impedir el acceso a personal no autorizado?	0	0	1	12	2	0	0	3	48	10
FIABILIDAD											
Madurez	¿El sistema funciona sin presentar fallas frecuentes?	0	0	5	10	0	0	0	15	40	0
Recuperabilidad	¿El sistema funciona sin perder datos por fallas ocasionales?	0	0	3	9	3	0	0	9	36	15
Tolerancia a fallos	¿Las fallas ocasionales del sistema no afectan gravemente el funcionamiento del sistema?	0	0	0	13	2	0	0	0	52	10
USABILIDAD											
Aprendizaje	¿Qué tan fácil es aprender a usar el sistema?	0	0	5	10	0	0	0	15	40	0
Comprensión	¿Qué tan fácil es comprender el funcionamiento del sistema?	0	0	8	7	0	0	0	24	28	0
Operatividad	¿El sistema puede ser operado sin mucho esfuerzo?	0	0	5	9	1	0	0	15	36	5
Atractividad	¿Qué tan atractiva es la apariencia del software?	0	0	5	10	0	0	0	15	40	0
EFICIENCIA											
Comportamiento en el tiempo	¿Qué tan rápido responde el sistema?	0	2	3	8	2	0	4	9	32	10
Comportamiento de recursos	¿El software, qué tan eficientemente maneja los recursos del sistema?	0	2	5	7	1	0	4	15	28	5
MANTENIBILIDAD											
Estabilidad	¿Qué tan estable es el software a pesar de los cambios?	0	0	4	8	3	0	0	12	32	15
Facilidad de análisis	¿Qué tan fácil es detectar las fallas en el software?	0	3	2	6	4	0	6	6	24	20
Facilidad de cambio	¿Qué tan fácil es modificar el software?	0	2	8	5	0	0	4	24	20	0
Facilidad de pruebas	¿Qué tan fácil es realizar pruebas en el sistema?	0	0	10	5	0	0	0	30	20	0
PORTABILIDAD											
Capacidad de instalación	¿Qué tan fácil es instalar el software?	0	0	2	10	3	0	0	6	40	15
Capacidad de reemplazamiento	¿Qué tan fácil es reemplazar el software por otro similar?	0	0	10	3	2	0	0	30	12	10
Adaptabilidad	¿Qué tan fácil es trasladar el software a otros ambientes?	0	5	8	2	0	0	10	24	8	0
Co-existencia	¿El software puede funcionar junto con otros sistemas?	0	0	5	9	1	0	0	15	36	5
TOTALES							**0**	**28**	**303**	**672**	**160**
PROMEDIO							**77.53**				

Fuente: Elaboración propia

ANEXO VI. Manual de usuario

SISTEMA PARA LA ADMINISTRACIÓN ACADÉMICA DEL CENTRO DE ESTUDIOS PREUNIVERSITARIOS DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO – PUNO

COMPONENTE DE ADMINISTRACIÓN

COMPONENTE DE ESTUDIANTES

COMPONENTE DE DOCENTES

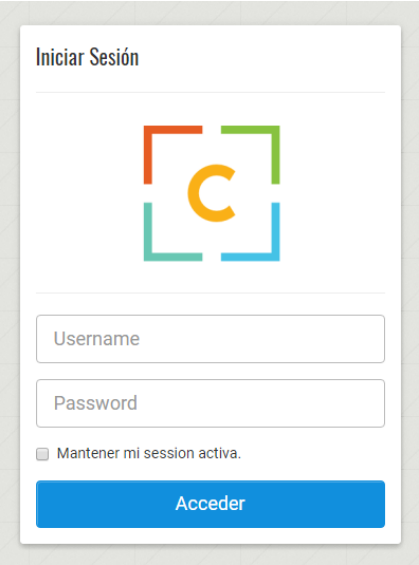
Elaborado por:

Efraín Ancachi Limachi

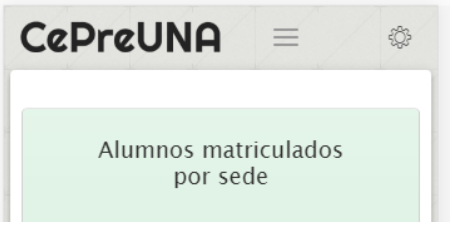
Febrero del 2017

1. ACCESO AL SISTEMA Y EL MÓDULO DE ADMINISTRACIÓN

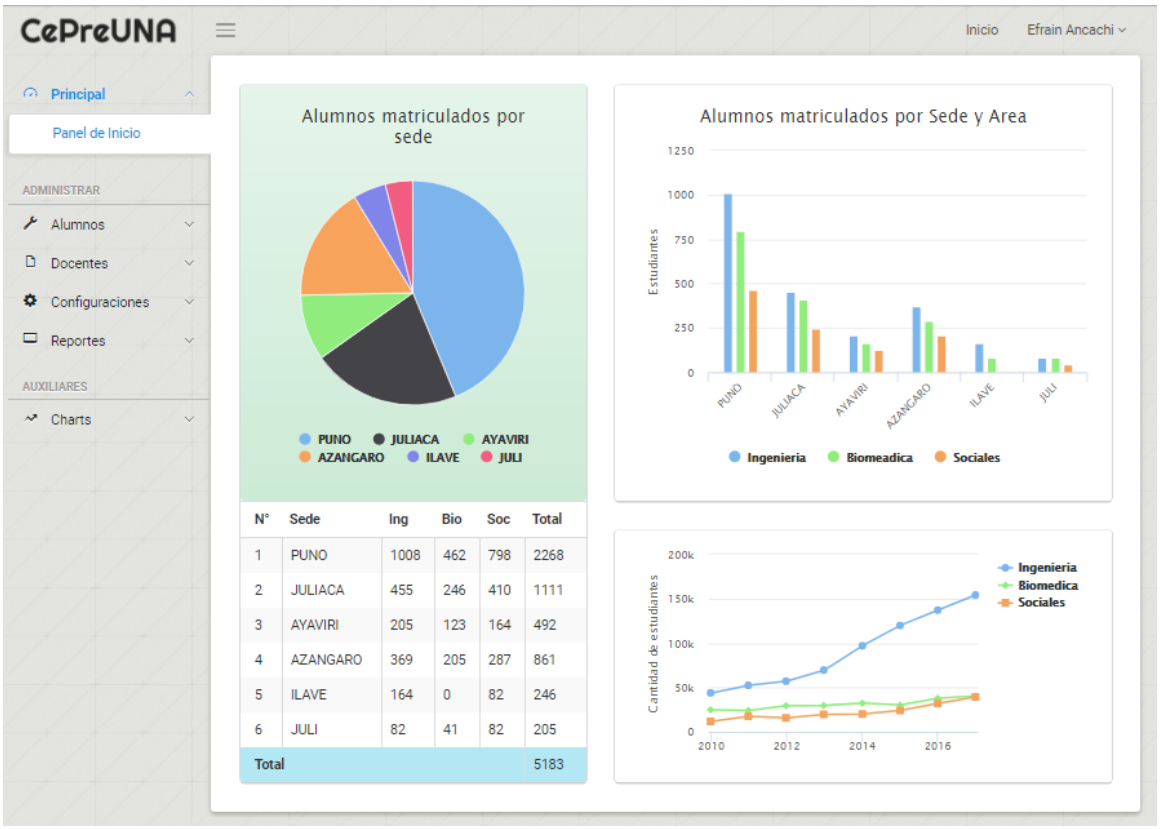
La primera pantalla que se pude observar al acceder al sistema, nos solicitará colocar los datos necesarios para permitirnos el acceso, si los datos introducidos fueran erróneos, esta mostrará un mensaje de alerta que nos indicara lo sucedido.



Una vez se haya superado el login o inicio de sesión, el sistema nos proporcionara acceso a la interfaz principal del módulo de administración, está conformada por una barra lateral que aloja al menú principal, un menú auxiliar ubicado en la parte superior de la pantalla y la zona de contenido, todo esto funciona también de manera adaptable (responsive), lo que nos permitirá trabajar en dispositivos móviles o PCs con pantallas de menor tamaño.



Vista del sistema en modo adaptado (responsive)

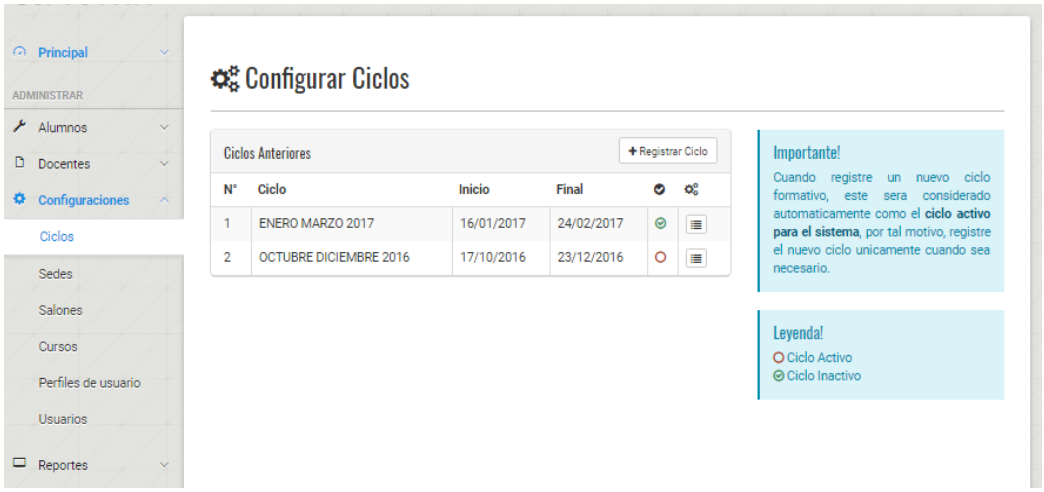


N°	Sede	Ing	Bio	Soc	Total
1	PUNO	1008	462	798	2268
2	JULIACA	455	246	410	1111
3	AYAVIRI	205	123	164	492
4	AZANGARO	369	205	287	861
5	ILAVE	164	0	82	246
6	JULI	82	41	82	205
Total					5183

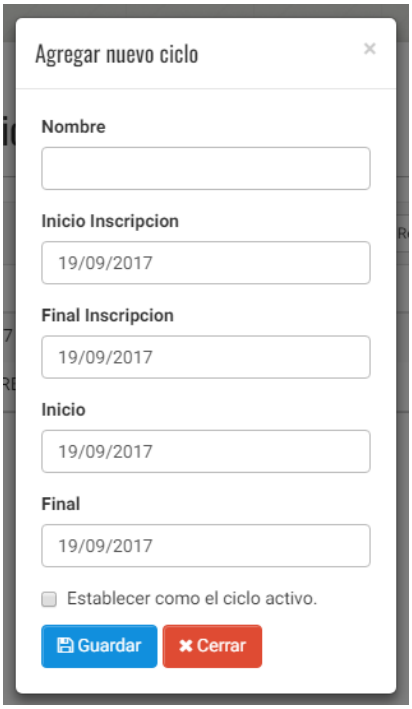
Vista de la pantalla principal del sistema

1.2. CONFIGURACION – Ciclos

Esta opción no permitirá administrar todos y cada uno de los ciclos formativos programados para cada año



Podemos agregar un nuevo ciclo haciendo clic en el botón “Registrar ciclo” que a su vez nos mostrara un ventana emergente cuya función es la de solicitar la información necesaria para la creación de un nuevo ciclo formativo.

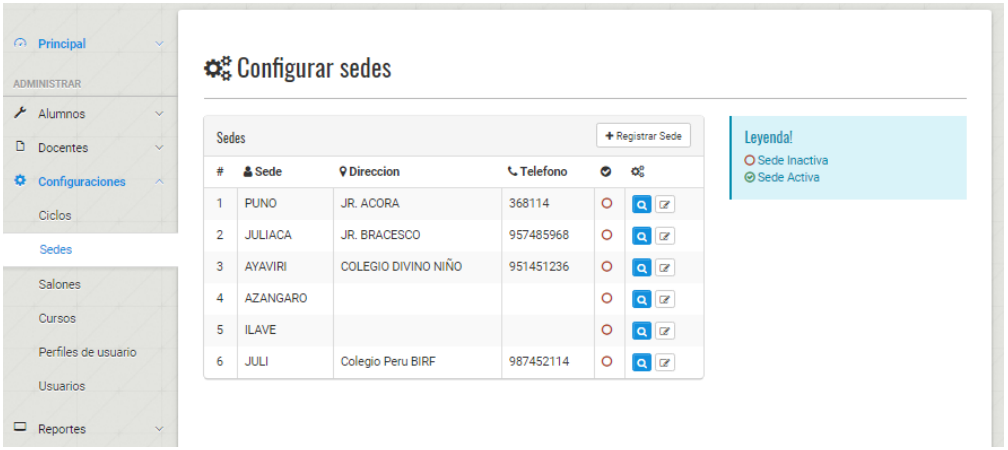


Cuando el sistema almacene la información, mostrara una alerta indicando que el proceso se ha completado exitosamente, caso contrario, nos mostrara una alerta que indicara la causa del problema.

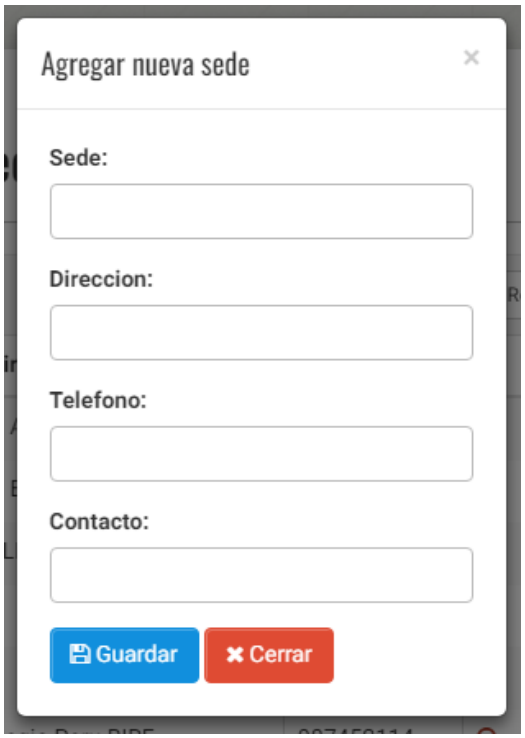
1.3. CONFIGURACION – Sedes

El submenú sedes del menú principal, nos dará el acceso a la sección que nos permitirá administrar y configurar las sedes con las que se contará por ciclo formativo, cada sede deberá contar con un coordinador.

Podemos hacer uso del botón “Registrar Sede”, para añadir nuevas sedes.



Cada registro cuenta con opciones independientes de edición y eliminación.



1.4. CONFIGURACION – Salones

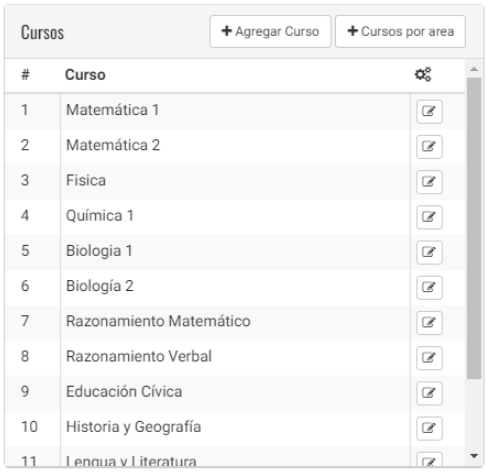
Esta opción del menú nos dará acceso a la interfaz para administrar el todos los salones por sede registrada.

Debemos hacer uso del formulario lateral dentro de esta sección para poder añadir nuevos salones, cada salón agregado será ubicado dentro de la pestaña que le corresponde según la sede que se seleccione.

Cada registro de salón, cuenta con opciones independientes para el acceso a la lista de estudiantes en formato PDF, y la visualización de información mucho más detallada.

1.5. CONFIGURACION – Cursos

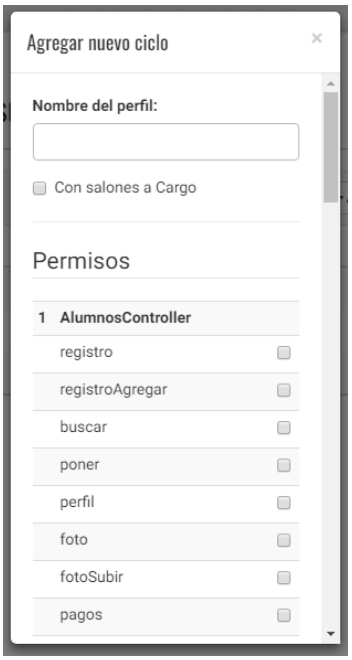
Esta sección nos mostrará dos paneles, el izquierdo para administrar cada curso de forma individual y el derecho que no permitirá la administración de los cursos por cada área formativa.



1.6. CONFIGURACION – Perfiles de usuario

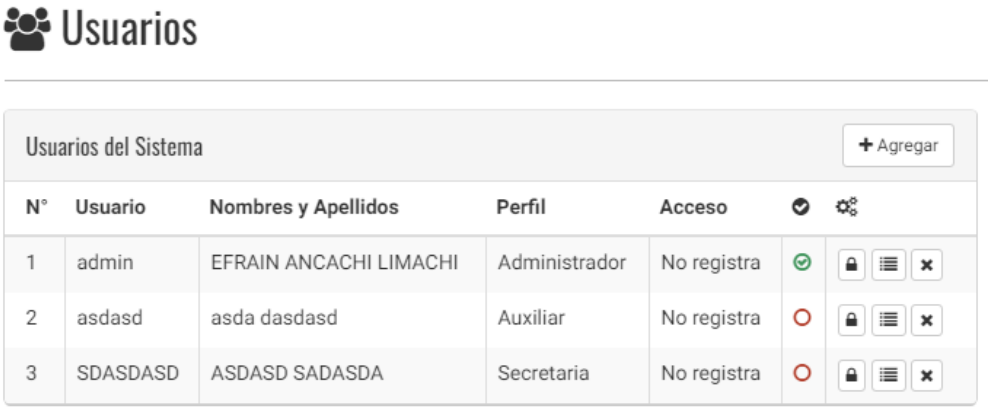
Administrar usuarios por perfiles nos permitirá asignarle a un grupo de usuarios los mismos permisos de acceso, para registrar un nuevo perfil, debemos hacer clic en el botón “Agregar perfil”:

El sistema solicitara la información necesaria para la creación del nuevo perfil, es importante también indicar que en este formulario deberá seleccionarse las rutas dentro del sistema a las que el usuario o los usuarios tendrán acceso.



1.7. CONFIGURACION - Usuarios

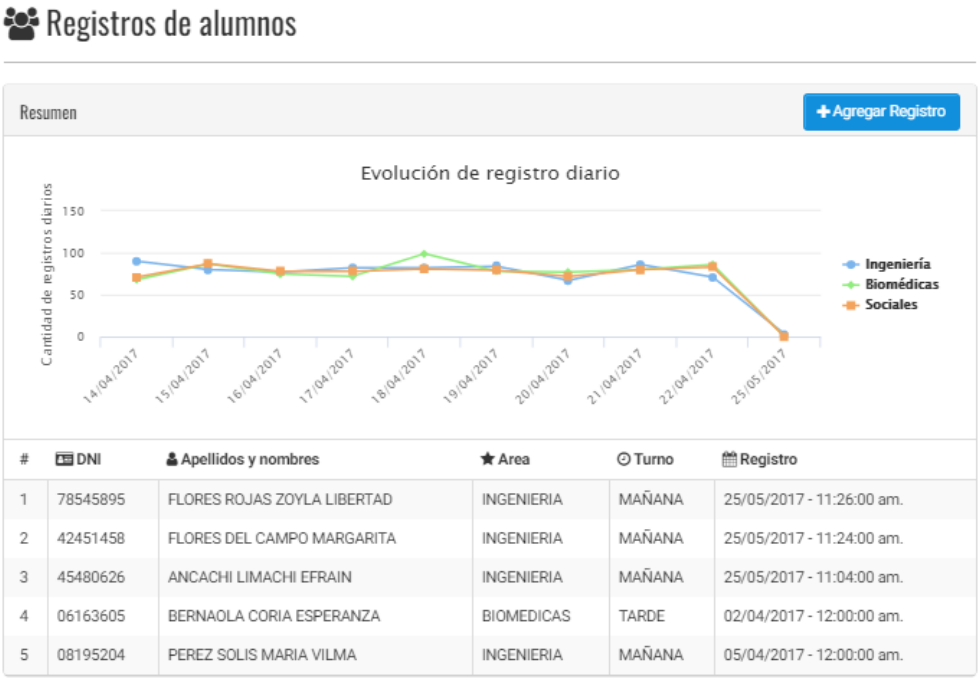
Esta sección nos muestra una lista de todos los usuarios del sistema, también nos detalla el perfil que tiene asignado y las acciones independientes que podemos realizar con cada uno de los registros.



Al hacer clic en el botón “Agregar” se visualiza una ventana emergente que no solicita la información necesaria para la creación del nuevo usuario.

1.8. ALUMNOS – Registros

Esta sección muestra, en gráfico, la cantidad de alumnos registrados por área durante los últimos 10 días, muestra también una pequeña lista con los últimos 5 registros realizados. Podemos también realizar un registro rápido haciendo clic en el botón “Agregar registro” que nos solicitará solo información básica y resumida para la creación del nuevo registro de estudiante a comparación del formulario principal.



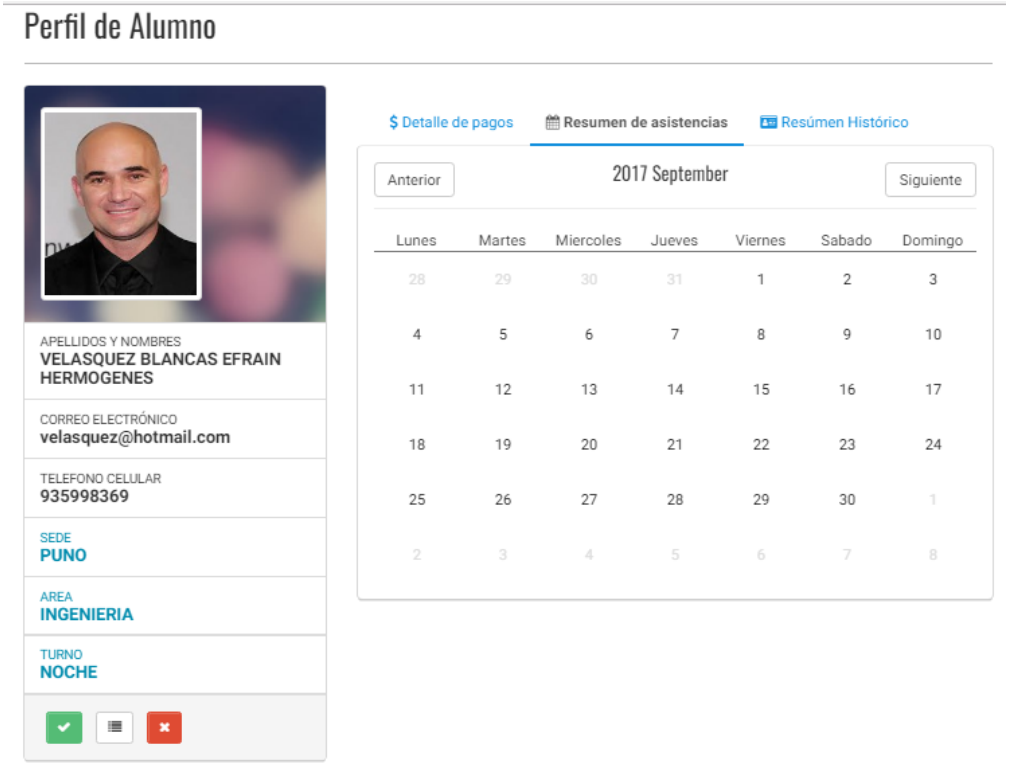
#	DNI	Apellidos y nombres	Area	Turno	Registro
1	78545895	FLORES ROJAS ZOYLA LIBERTAD	INGENIERIA	MAÑANA	25/05/2017 - 11:26:00 am.
2	42451458	FLORES DEL CAMPO MARGARITA	INGENIERIA	MAÑANA	25/05/2017 - 11:24:00 am.
3	45480626	ANCACHI LIMACHI EFRAIN	INGENIERIA	MAÑANA	25/05/2017 - 11:04:00 am.
4	06163605	BERNAOLA CORIA ESPERANZA	BIOMEDICAS	TARDE	02/04/2017 - 12:00:00 am.
5	08195204	PEREZ SOLIS MARIA VILMA	INGENIERIA	MAÑANA	05/04/2017 - 12:00:00 am.

La información solicitada por este pequeño formulario obviamente se encontrará incompleta en la base de datos, por tal razón el estudiante cuyo registro haya sido creado por este formulario deberá completar la información restante la primera vez que inicie sesión en el sistema, de tal manera se asegurara la integridad de los datos. La falta de información en cada registro podría acarrear errores en los sistemas derivados de este, por tal razón es necesaria brindar las indicaciones necesarias si se va a hacer uso de este modo rápido de registro.

1.9. ALUMNOS – Matricular

Al hacer clic sobre esta opción, lo primero que el sistema muestra es una ventana emergente de búsqueda, en la que se debe colocar algún parámetro para realizar la búsqueda, ya sea el N° de documento, apellidos o nombre, una vez realizada la búsqueda, se debe hacer clic en el botón “Colocar” que tiene cada registro, esto para colocar el registro seleccionado en la interfaz de perfil.

Una vez cargado el registro en el interfaz de perfil, se podrá tener información detallada del registro, esto incluye matricula, pagos, asistencias, información histórica,



Es importante mencionar que es en esta sección en la que deberá realizarse la matrícula presionando el botón , si la matricula ya ha sido realizada, al presionar este botón, se nos dará los detalles de la matrícula.

2. COMPONENTE DE ESTUDIANTES

2.1. Registro de alumnos

Todos los estudiantes, para iniciar su registro deben acceder a la dirección proporcionada por el administrador del sistema para acceder al formulario de registro de alumnos.

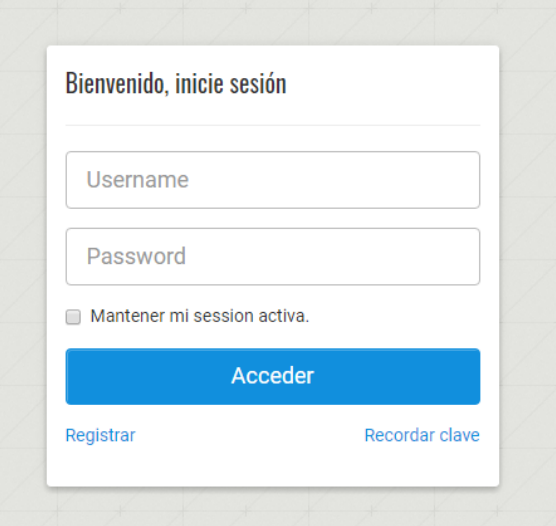
Este formulario solicitara toda la información que el sistema requiere para un correcto funcionamiento, cada sección cuenta con un instructivo que orienta al estudiante para que pueda realizar su registro de manera correcta para así poder evitar los problemas posteriores que pueda causar el llenado incompleto del formulario.

Una vez completado llenado el formulario con la información necesaria, se debe hacer clic en el botón “Registrar” para enviar toda la información al servidor.

2.2. Acceso al módulo de estudiantes

Posterior al registro, para poder acceder al módulo de estudiantes, el usuario ha debido validar su inscripción de manera presencial en el CEPREUNA.

Para acceder a este módulo se deberá acceder a través de la dirección proporcionada por el administrador del sistema, la pantalla de logueo debería presentarse tal y como se muestra en la siguiente imagen.



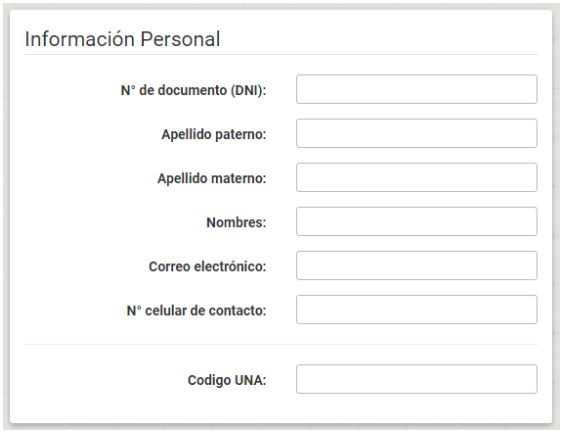
Si el usuario supera la pantalla de login proporcionando sus credenciales, el sistema le mostrará la siguiente interfaz:

Este interfaz proporciona al alumno acceso únicamente a la información más relevante para que este pueda estar al pendiente del registro de sus asistencias y pagos.

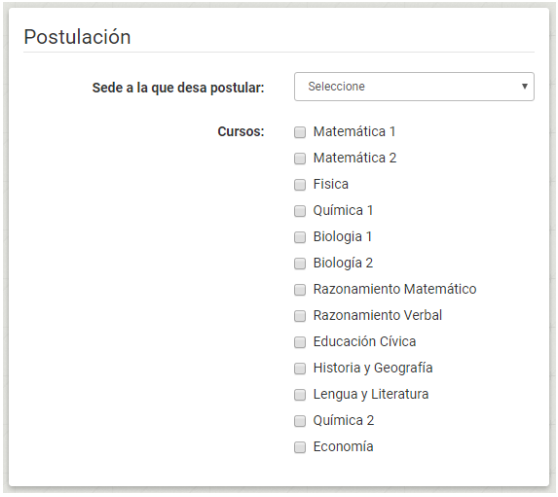
3. COMPONENTE DE DOCENTES

3.1. Registro de docentes

Para el registro de docentes, este debe de acceder a la dirección proporcionada por el administrador del sistema y completar el siguiente formulario:



En esta sección el docente deberá proporcionar su información personal.



Aquí el docente deberá registrar la sede a la que desea postular y los cursos que pretende desarrollar.

En esta sección el docente debe colocar su disponibilidad de tiempo para que el personal administrativo pueda tomarlo en cuenta al momento de la selección.

Una vez completada la información solicitada se debe proceder a hacer clic en el botón “Guardar Registro”, esto almacenara la información en la base de datos y se mostrara al usuario la siguiente pantalla que indicará que el registro ha sido exitoso:

El docente deberá imprimir su ficha de postulación haciendo clic en el botón “Imprimir ficha”, para finalizar el registro se debe presionar el botón “Terminar Inscripción”: