ESCUELA DE INGENIERÍA EN SISTEMAS

Tema:

"IMPLEMENTACIÓN DE UN SERVICIO DE NOTIFICACIÓN DEL REGISTRO DE ASISTENCIA PARA LOS TRABAJADORES DE LA PUCESA"

Disertación de grado previa a la obtención del título de Ingeniero de Sistemas y Computación

Línea de Investigación:

Ingeniería de Software (Arquitectura y procesos)

Autor:

GABRIEL EDUARDO ALTAMIRANO IBARRA

Director:

Ing. Mg. DARÍO JAVIER ROBAYO JÁCOME

Ambato – Ecuador

Diciembre 2014

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

SEDE AMBATO

HOJA DE APROBACIÓN

Tema:

"IMPLEMENTACIÓN DE UN SERVICIO DE NOTIFICACIÓN DEL REGISTRO DE ASISTENCIA PARA LOS TRABAJADORES DE LA PUCESA"

Línea de Investigación:

Ingeniería de Software (Arquitectura y procesos)

Autor:

GABRIEL EDUARDO ALTAMIRANO IBARRA

DARÍO JAVIER ROBAYO JÁCOME, Ing. Mg. f._____

CALIFICADOR

TERESA MILENA FREIRE AILLÓN, Ing. Mg. f._____

CALIFICADOR

ENRIQUE XAVIER GARCÉS FREIRE, Ing. Mg. f._____

CALIFICADOR

GALO MAURICIO LÓPEZ SEVILLA, Ing. Mg. f._____

DIRECTOR DE LA ESCUELA DE INGENIERÍA EN SISTEMAS

HUGO ROGELIO ALTAMIRANO VILLARROEL, Dr. f._____

SECRETARIO GENERAL PUCESA

Ambato – Ecuador

Diciembre 2014

DECLARACIÓN DE AUTENTICIDAD

Y RESPONSABILIDAD

Yo, Gabriel Eduardo Altamirano Ibarra portador de la cédula de ciudadanía No. 180402530-0 declaro que los resultados obtenidos en la investigación que presento como informe final, previo la obtención del título de INGENIERO DE SISTEMAS Y COMPUTACIÓN son absolutamente originales, auténticos y personales.

En tal virtud, declaro que el contenido, las conclusiones y los efectos legales y académicos que se desprenden del trabajo propuesto de investigación y luego de la redacción de este documento son y serán de mi sola y exclusiva responsabilidad legal y académica.

Gabriel Eduardo Altamirano Ibarra

CI. 180402530-0

AGRADECIMIENTO

A Dios por su infinito amor.

A mis Padres y mi Hermano por su apoyo incondicional.

Al Director y Lectores de este trabajo por su valiosa guía.

Al Prorrector, Director de Talento Humano y quienes conforman el Departamento de Informática de la Pontificia Universidad Católica del Ecuador Sede Ambato por su valiosa colaboración.

Así también al Director y Docentes de la Escuela de Ingeniería en Sistemas de la Pontificia Universidad Católica del Ecuador Sede Ambato por sus enseñanzas.

Gabriel.

DEDICATORIA

A mi madre.

Gabriel.

RESUMEN

Este documento muestra la implementación de un servicio de notificación del registro de asistencia que propende a reducir la incertidumbre como riesgo social en los trabajadores de la PUCESA lo que evita favorecer el presentismo laboral y mejorar el desempeño del trabajador y la organización. Este servicio consiste en notificar mediante correo electrónico institucional los inadecuados registros de asistencia. Las notificaciones pueden ser consultadas a través de una aplicación web. El servicio es implementado en Windows Server 2008 R2 SP1 que procesa la información tomada del lector biométrico Hand Punch 2000 a través de un convertidor de serie a Ethernet, DS Manager y el software HP32. Debido a que el web service otorgado por la PUCESA para acceso a la información usa tecnología SOAP el consumo del mismo se lo realiza a través de clases nativas de Java lo que determina que el desarrollo se lo realice en lenguaje jRuby sobre el servidor de aplicaciones jBoss a través de Torquebox lo cual soporta al framework Ruby on Rails que también es usado como parte de la solución integrando varias gemas. La metodología usada para el desarrollo es “Extreme Programming” la cual es ágil y se caracteriza por valorar a los individuos y sus interacciones, funcionabilidad y reacción ante el cambio.

ABSTRACT

This paper shows the implementation of a notification facility of the attendance records that tends to reduce uncertainty as a social risk within the people who work at PUCESA; avoiding presenteeism and improving the employees and organization performances. This facility notifies absences or delays records by using institutional email and could be found using a web application. The service is deployed on Windows Server 2008 R2 SP1 that processes information taken from the Hand Punch 2000 biometric reader by using a serial to the Ethernet converter, DS Manager and the HP32 software. Because of the web service provided by PUCESA to access to the information, it uses SOAP technology. The consumption is proccesed by native a Java class which determines that the development applies jRuby language on the jBoss application server through Torquebox which will support the Ruby on Rails framework, also used as a part of the solution integrating several gems. The methodology used for the development is 'Extreme Programming' which is fast and characterized by valuing individuals and interactions, functionality and the reaction to the change.

ÍNDICE DE CONTENIDOS

ÍNDICE DE GRÁFICOS

Ilustraciones

Tablas

INTRODUCCIÓN

En el presente trabajo, disertación de grado previa a la obtención del título de Ingeniero de Sistemas y Computación titulada “IMPLEMENTACIÓN DE UN SERVICIO DE NOTIFICACIÓN DEL REGISTRO DE ASISTENCIA PARA LOS TRABAJADORES DE LA PUCESA“, se desarrolla la infraestructura tecnológica necesaria para la ejecución de la misma.

En el primer capítulo se describe al problema que dio origen este proyecto tomando a la Dirección de Talento Humano de la PUCESA como punto central en la ejecución del mismo, específicamente el registro de asistencia y la incertidumbre generada en este proceso. Así también se justifican las razones por las que el problema debe ser abordado y los objetivos que se alcanzan. Finalmente fundamenta teórica y científicamente el servicio a implementar.

En el segundo capítulo se estudia a Extreme Programming – XP ó Programación Extrema como metodología de desarrollo a través de: valores, reglas y sus fases como planificación, diseño, implementación (codificación) y pruebas.

El tercer capítulo evidencia el desarrollo de la metodología estudiada anteriormente, documentando las actividades realizadas a lo largo del trabajo, desde la planificación hasta las pruebas de XP, se añade el despliegue de la aplicación como soporte para mantener eficazmente el servicio.

El cuarto capítulo refiere el análisis de resultados, demostrando el servicio implementado y en producción a través de sus múltiples funcionalidades, siendo validadas por su principal beneficiario, el Director de Talento Humano de la PUCESA.

Finalmente se puntualizan las conclusiones y recomendaciones de la ejecución de este proyecto. Esta disertación referencia la bibliografía utilizada y un anexo que propende al correcto uso del servicio.

CAPÍTULO I
FUNDAMENTOS TEÓRICOS

1.1 Antecedentes

La Pontificia Universidad Católica del Ecuador Sede Ambato (PUCESA), es una institución de educación superior que trabaja continuamente en la formación de profesionales. En esta comunidad universitaria los trabajadores se clasifican en docentes y administrativos.

La PUCESA como organización tiene constituido el Departamento de Talento Humano que tiene como competencia la gestión del personal que labora en la mencionada institución.

El Ministerio de Relaciones Laborales de la República del Ecuador precautela la integridad de condiciones laborales con las que trabajan millones de ecuatorianos en las diversas organizaciones constituidas, así también controla que los trabajadores cumplan con lo acordado en

los diferentes contratos que han sido debidamente legalizados.

La diversidad de contratos con los que cuenta la legislación ecuatoriana es extensa, por lo que actualmente una de las variables que debe ser controlada es la de los horarios de trabajo. Con este objetivo, las organizaciones han tomado medidas que armonizan los derechos y obligaciones de cada uno de los trabajadores en torno a este tema.

Entre las múltiples soluciones implementadas, se puede observar el uso común de sistemas biométricos con los que se controla el registro en los horarios acordados con cada uno de los trabajadores en sus contratos, de tal manera que esta información sirva para la elaboración de roles de pago.

Una de las principales razones por las que el uso de sistemas biométricos se ha proliferado, es la reducción de fraude en el registro de los trabajadores, así como controlar la ausencia o retraso de los mismos. En muchos de los casos las faltas injustificadas son tomadas en cuenta por el Departamento de Talento Humano y reportadas

a jefes inmediatos que optan por el descuento directo en la remuneración, llamados de atención u otros mecanismos de corrección.

El correo electrónico institucional de la PUCESA ha venido siendo subutilizado únicamente para comunicación individual o masiva. En este caso se puede usar como medio de notificación para minimizar la incertidumbre que afecta a la mayoría de colaboradores de la PUCESA respondiendo a la interrogante de si se registró o no, emitiendo reportes diarios de tal forma que se puedan evitar sanciones económicas o llamados de atención sin fundamento.

1.2 Problema

1.2.1 Descripción del Problema

La notificación del registro de asistencia para los trabajadores de la PUCESA, que permita reducir la incertidumbre de haber registrado o no en los horarios acordados, ha venido siendo una necesidad para diversas

unidades de la institución, pues existen sanciones económicas o llamadas de atención frecuentes al personal.

En los últimos años, se ha evidenciado un incremento significativo de métodos de control de asistencia del personal de una organización, especialmente los sistemas biométricos que se encuentran en el mercado en diversos modelos.

El contar con un servicio de notificación del registro de asistencia para los trabajadores permitiría precautelar la integridad laboral de los miembros de una organización, así como también ahorrar tiempo en verificaciones manuales del registro de cada uno de los colaboradores.

Así, la PUCESA ha estimado disminuir ésta problemática notificando el registro inapropiado, de tal forma que el empleado al igual que la Dirección de Talento Humano estén en conocimiento de estos eventos; todo esto aprovechando herramientas que no han sido explotadas para la administración de la universidad e infraestructura tecnológica disponible.

1.2.2 Preguntas Básicas

- ¿De qué manera la PUCESA notifica el registro de asistencia a los trabajadores de la PUCESA?
- ¿Cuáles son los beneficios para la Dirección de Talento Humano y los trabajadores al contar con un servicio de notificación del registro de asistencia?
- ¿Qué herramientas y técnicas son necesarias para el desarrollo de un servicio de notificación del registro de asistencia para los trabajadores de la PUCESA?
- ¿Cuáles deberían ser las políticas y funciones para el uso del servicio de notificación del registro de asistencia para los trabajadores de la PUCESA?

1.3 Justificación

Los sistemas biométricos de control han tomado un papel protagónico en la administración de una organización. Los encargados de la gestión del Talento Humano en las organizaciones lo utilizan como la principal referencia del cumplimiento de las responsabilidades acordadas en cuanto a horarios respecta y en muchos de los casos la base fundamental para la elaboración de roles de pago.

En base a ello, las organizaciones toman correctivos sobre las faltas que registran los empleados, medidas que van desde llamados de atención verbales o escritos hasta sanciones económicas. En muchos casos las sanciones impuestas pueden no llegar a tener fundamento de incumplimiento, sino de una falta involuntaria por olvido.

La PUCESA consiente de este problema y siendo consecuente con su filosofía Cristiana pretende disminuir las sanciones económicas o llamados de atención a sus colaboradores implementando un sistema de notificación del registro de asistencia, de tal manera que se prevea cualquier sanción infundada.

Los principales beneficiarios de este servicio serán los trabajadores de la PUCESA y la Dirección de Talento Humano, ya que luego de la implementación de este servicio los reclamos de no conformidad en los roles de pago serán escasos, debido a que, cada empleado tendrá a su disposición información real y a tiempo sobre sus registros.

El tema de desarrollo resulta novedoso por cuanto no existen trabajos similares en este campo dentro de la PUCESA. Muy pocas organizaciones han logrado notificar a tiempo correctamente del registro a cada empleado, por lo que, es un campo que no ha sido explotado.

1.3.1 Justificación Técnica

La implementación del presente servicio de notificación del registro de asistencia para los trabajadores de la PUCESA involucra todos los conocimientos adquiridos a lo largo de la formación como Ingeniero de Sistemas y Computación de la Pontificia Universidad Católica del Ecuador Sede Ambato, por lo que se justifica la capacidad de ejecución del mismo tanto personal como profesional.

1.3.2 Justificación Económica

La implementación de un servicio de notificación del registro de asistencia para los trabajadores de la PUCESA cuenta con los recursos necesarios para su elaboración.

1.4 Objetivos

1.4.1 Objetivo General

Implementar un servicio de notificación del registro de asistencia para los trabajadores de la PUCESA.

1.4.2 Objetivos Específicos

- Fundamentar teórica y científicamente acerca de servicios de notificación del registro de asistencia.
- Diagnosticar la forma en que la PUCESA notifica el registro de asistencia a sus colaboradores.
- Desarrollar el sistema de notificación del registro de asistencia para los trabajadores de la PUCESA utilizando Ruby on Rails.

1.5 Pregunta de Estudio

¿Cómo notificar a los trabajadores de la PUCESA el inadecuado registro de asistencia?

1.6 Fundamentos Teóricos

1.6.1 Biometría

Según Jain et al. (2008), la Biometría es la ciencia de reconocimiento de identidad de una persona basado en los atributos físicos y de comportamiento del individuo como su rostro, huellas dactilares, voz e iris. Así, la gran necesidad de una técnica robusta de reconocimiento humano en aplicaciones críticas como control de acceso, cruce de fronteras internacionales y cumplimiento de la ley se ha posesionado por sí misma como una solución viable que puede ser integrada en sistemas de gestión de identidad a gran escala.

Los sistemas biométricos funcionan bajo la premisa que muchas de las características físicas o de comportamiento de los humanos son distintivas, mismas que pueden ser adquiridas por sensores y representados en formato numérico que permitirá tomar decisiones automáticas acerca de la identidad.

La biometría para Jain et al. (2008), ofrece una solución natural y segura para ciertos aspectos de la gestión de identidad por la utilización de esquemas totalmente automatizados o semi-automatizados para reconocer individuos en base a sus características biológicas. Por lo que usando biometría es posible establecer una identidad basado en ¿quién eres? a diferencia de ¿por qué posees?, como una tarjeta de identificación, o ¿qué recuerdas? como una clave. En algunas aplicaciones, la biometría puede ser usado como suplemento a tokens y contraseñas, de esta forma se añade un nivel de seguridad (véase Ilustración 1.1).

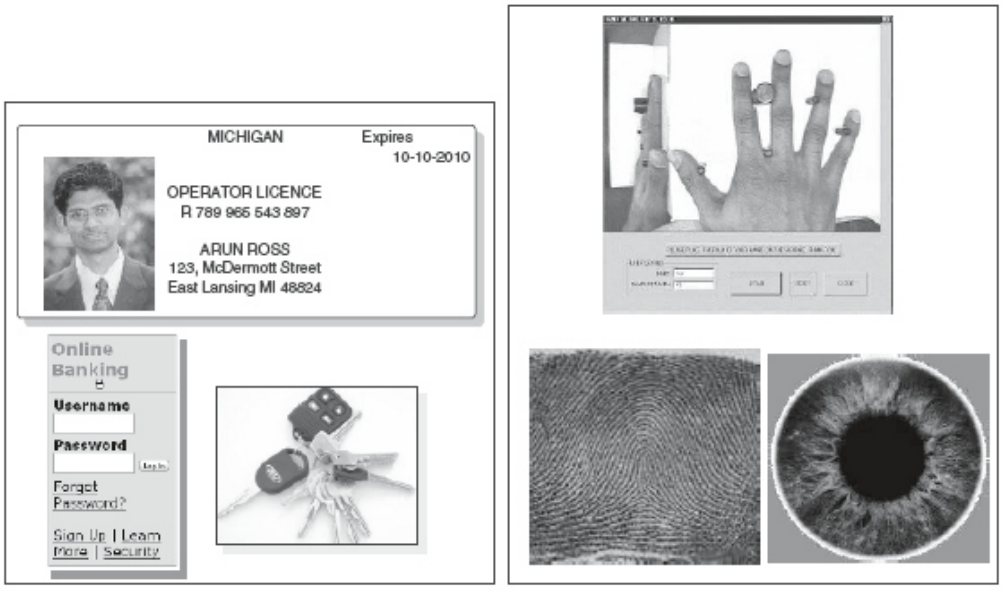


Ilustración 1.1: Esquemas de Autenticación.

Fuente: Handbook of Biometrics, 2008

Los sistemas biométricos utilizan una variedad de características físicas o de comportamiento que incluyen la huella dactilar, rostro, geometría de las manos, iris, retina, firma, forma de caminar, huella palmar, patrón de voz, oído, venas de las manos, olor o ADN (véase Ilustración 1.2). Jain et al. (2008), afirma que mientras los sistemas biométricos tienen sus propias limitaciones, tienen una ventaja sobre los métodos de seguridad tradicionales en que no pueden ser fácilmente robados o compartidos. Además que mejoran la comodidad del usuario al aliviar la necesidad de diseñar y recordar contraseñas.

Ilustración 1.2: Ejemplos de Rasgos Biométricos.

Fuente: Handbook of Biometrics, 2008

1.6.1.1 Operación de un Sistema Biométrico

Un sistema biométrico es esencialmente un sistema de reconocimiento de patrones que adquiere información biométrica de un individuo, extrae un conjunto de características sobresalientes, compara estas características con las almacenadas en la base de datos, y ejecuta una acción basada en los resultados de la comparación. Así, un sistema biométrico genérico puede ser compuesto según Jain et al. (2008), por cinco módulos principales: sensor, evaluación de calidad, extracción de características, comparación y base de datos.

El módulo sensor es requerido para adquirir la información biométrica no procesada de un individuo, define la interfaz de máquina humana, mismo que es crucial para el rendimiento del sistema biométrico. Una pobre interfaz puede desembocar en un fracaso y en consecuencia una baja aceptabilidad del usuario. La calidad de extracción de información biométrica es primero evaluado por el sensor para determinar su idoneidad para futuros procesos. Generalmente los datos adquiridos se someten a un algoritmo de mejora de la señal con el fin de optimizar su calidad. Sin embargo, en

algunos casos, la calidad de la información biométrica puede no ser la esperada así que se le pedirá al usuario presentar los datos biométricos de nuevo. Los datos biométricos se procesan y un conjunto de características discriminatorias sobresalientes se extraen para representar el rasgo subyacente. El módulo de comparación confronta las características extraídas con las de plantillas almacenadas para generar puntuaciones de correspondencia y también encapsula una decisión ya que los resultados de las comparaciones se utilizan para validar, ya sea una identidad declarada o proporcionar un ranking de las identidades inscritas en orden para identificar a un individuo. Por otro lado el módulo de la base de datos del sistema actúa como un repositorio de información biométrica ya que durante el proceso de inscripción, un conjunto de características son extraídas para la muestra biométrica en bruto (es decir, la plantilla) que se almacena en la base de datos, posiblemente junto con cierta información biográfica como nombre, número de identificación personal (PIN), dirección, etc, que caracterizan el usuario. La plantilla de un usuario puede ser extraída a partir de una única muestra biométrica, o generada mediante el procesamiento de múltiples muestras; algunos sistemas almacenan varias plantillas con el fin de dar cuenta de las variaciones asociadas con un usuario; dependiente de la aplicación,

la plantilla se puede almacenar en la base de datos central del sistema biométrico o grabarse en un token (por ejemplo, tarjetas inteligentes) emitidos al individuo. Finalmente dependiendo del contexto de aplicación, un sistema biométrico puede operar ya sea en el modo verificación o identificación.

Según O' Gorman (2003), en el modo verificación, el sistema valida la identidad de una persona mediante la comparación de los datos biométricos capturados con su propia plantilla biométrica almacenada en la base de datos del sistema. En tal sistema, un individuo que desea ser reconocido afirma una identidad, por lo general a través de un PIN, un nombre de usuario o una tarjeta inteligente, la verificación se utiliza normalmente para reconocimiento positivo, en el que el objetivo es impedir que varias personas utilicen la misma identidad.

En el modo de identificación, el sistema reconoce a un individuo con la búsqueda de plantillas de todos los usuarios en la base de datos para una coincidencia, por lo tanto, el sistema lleva a cabo una comparación de uno a muchos para establecer una identidad individual o

falla, si el sujeto no está inscrito en la base de datos del sistema.

1.6.1.2 Uso de Características Biométricas

Según Jain et al. (2008), la identificación utilizando huellas dactilares ha demostrado ser muy efectiva ya que una huella digital es el patrón de crestas y valles en la superficie de la yema del dedo cuya formación se determina en los primeros siete meses de desarrollo fetal, pero las huellas dactilares de una pequeña fracción de la población pueden ser inadecuadas para la identificación automática debido a factores genéticos, envejecimiento, razones ambientales u ocupacionales (por ejemplo, los trabajadores manuales pueden tener un gran número de cortes y moretones en sus huellas digitales que las mantienen cambiantes). Así mismo la geometría de la mano se basa en un número de mediciones tomadas de la mano humana, incluyendo su forma, tamaño de la palma, las longitudes y anchuras de los dedos. Basado en Jain et al. (2008), las huellas palmares contienen un patrón de crestas y valles muy parecidos a las huellas dactilares, el área de la palma es mucho más grande que el área de un dedo y como resultado se espera que las huellas palmares

sean aún más distintivas que las huellas dactilares. Los escáneres palmares necesitan capturar una gran superficie, por lo que son más voluminosos y más caros que los sensores de huellas dactilares, las palmas humanas también contienen un distintivo adicional tales como líneas principales y las arrugas que se pueden capturar incluso con un escáner de resolución más baja, lo que sería más barato.

1.6.1.3 HandPunch 2000

El HandPunch 2000 (véase Ilustración 1.3) brinda exactitud y conveniencia de la tecnología biométrica en aplicaciones de asistencia. Es un miembro de la línea de reconocimiento de la geometría de la mano, el HandPunch 2000 registra y almacena la forma tridimensional de la mano humana para la comparación y verificación de identidad. Tras la verificación, el HandPunch 2000 registra el tiempo, fecha, número de identificación del usuario, el tiempo de recogida de datos y la asistencia para su recogida por un ordenador central.

Ilustración 1.3: HandPunch® 2000

Fuente: HandPunch 2000: Manual, 2003

1.6.1.3.1 HandPunch 2000: Características

- **Número de pieza:** HP-2000
- **Tamaño:** 8.85 in (22,3 cm) de ancho 11.65 in (29,6 cm) de alto 8.55 (21,7 cm) de profundidad
- **Peso:** 6 libras (2,7 kg)
- **Alimentación:** 12 a 24 VDC o 12 a 24 VAC 50-60 Hz, 7 vatios

- **Rango de temperatura:** 14 °F a 140 °F (-10 °C a 60 °C) sin funcionar/almacenar; 32 °F a 113 °F (0 °C a 45 °C) en funcionamiento
- **Humedad relativa:** (sin condensación) 5% a 85% no operativo; 20% a 80% operativo.
- **Tiempo de verificación:** Menos de un segundo
- **Retención de la memoria:** hasta 5 años a través de la batería de litio interna estándar.
- **Capacidad de registros:** 5120 transacciones
- **Longitud de número de identificación:** 1 a 10 dígitos desde el teclado o la tarjeta.
- **Capacidad del usuario:** 512 usuarios
- **Tamaño de la Plantilla:** 9 bytes
- **Comunicaciones:** RS-232, cable de 50 pies incluido.
- **Velocidad:** 300 a 28,8 kbps
- **Opciones:** BB-200 - Batería de reserva operacional; MB-500 - módem interno de alta velocidad (Recognition Systems, 2003)

1.6.1.3.2 HandPunch 2000: Principio de Operación.

Según Recognition Systems (2003), el HandPunch 2000 utiliza la luz infrarroja de bajo nivel, la óptica y un CMOS (chip IC) cámara para capturar una imagen

tridimensional de la mano. Gracias a la avanzada tecnología del microprocesador, el HandPunch 2000 convierte la imagen a una plantilla electrónica misma que es almacenada en una base de datos con el número de identificación del usuario. Para registrarse, el usuario introduce su número de identificación en el teclado del HandPunch 2000 o utiliza un lector de tarjetas externo, luego se pide al usuario que coloque su mano en el cristal de exposición (véase Ilustración 1.4). El HandPunch 2000 compara la mano en la platina con la plantilla única del usuario. Si las imágenes coinciden, registra la transacción para su procesamiento.

Ilustración 1.4: Vista Frontal del HandPunch® 2000

Fuente: HandPunch 2000: Manual, 2003

A continuación se muestran las conexiones que el HandPunch 2000 ofrece en su parte posterior (véase Ilustración 1.5), así también la configuración con la que transmite información por RS-232 (véase Tabla 1.1).

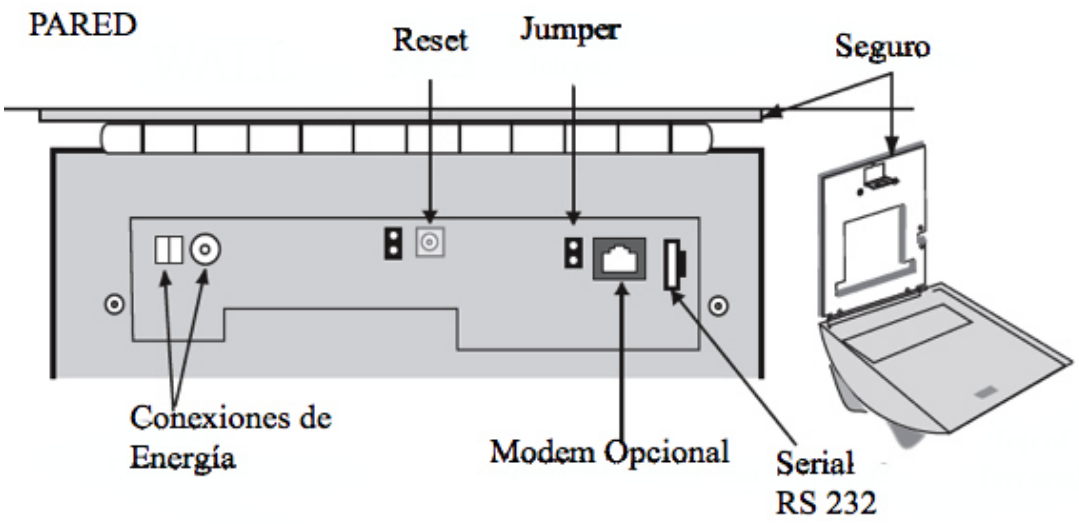


Ilustración 1.5: Vista Posterior del HandPunch® 2000

Fuente: HandPunch 2000: Manual, 2003

J8 Pin	Signal	Connection
1	GND	Ground
2	RXD	Receive Data Input (from external device)
3	TXD	Transmit Data Output (to external device)
4	RTS	Ready to Send Output (to external device)

Tabla 1.1: Conexión RS-232 Serial

Fuente: HandPunch 2000: Manual, 2003

1.6.1.4 Dispositivo Serial Servidor DS100.

El DS100 (véase Ilustración 1.6) es un servidor original de Tibbo (convertidor de serie a Ethernet). El DS100 conecta externamente prácticamente cualquier dispositivo serie existente a una red Ethernet.

El DS100 está disponible en dos versiones: DS100R con puerto serie RS232 y DS100B con puerto serial RS232/422/485 universal.

Ilustración 1.6: Dispositivo Serial Servidor DS100.

Fuente: Tibbo Technology, 2009

1.6.1.4.1 Especificaciones Técnicas

- Basado en el módulo EM100 Ethernet.

- Puerto Ethernet 10BaseT.
- DS100R: un puerto serie RS232 (TX, RX, RTS, CTS líneas).
- DS100B: puerto RS232/422/485 universales:
    - Modo RS232: RX, TX, RTS, CTS, DTR, DSR líneas;
    - RS422: RX + / -, TX + / -, CTS + / -, RTS + / - líneas;
    - RS485: RX + / -, TX + / -, el control automático de dirección.
- Cuatro LEDs de estado.
- Potencia: 9-16V (12V nominal).
- Dimensiones: 89x51x30mm.
- El firmware es actualizable a través del puerto serie o de red.
- CE (DS100 y DS100B) y FCC (DS100 y DS100B) certificados. (Tibbo Technology,2009)

Funcionalidad

- Modos de puerto de serie completo y half-duplex.
- De red del servidor, cliente y servidor / cliente ("routing") los modos.
- Otras numerosas opciones para las conexiones, puerto serie, etc

- Configuración almacenados en la EEPROM.
- Indicación de estado detallada a través de LEDs.
- Configuración a través del puerto serie o de red (UDP).
- Control remoto de RTS, CTS, DTR, y las líneas DSR.
- "On-the-fly" comandos para la configuración del puerto serie inmediata.
- Del lado de serie "módem" comandos para el control de las conexiones de red.
- Control directo de los módems ADSL.
- Soporta UDP, TCP, ARP, ICMP (ping), DHCP, PPPoE, LCP. (Tibbo Technology,2009)

1.6.1.5 HandPunch Software.

El HandPunch-32 (HP-32) es una utilidad de software que permite la administración y la integración de terminales de lectura de mano con software de asistencia. Esto se logra mediante la recuperación de datos de los terminales en el sistema. Los datos recibidos desde los terminales están en formato ASCII sencillo con un registro por línea, que luego pueden ser procesados por un software de terceros.

Según Amano Cincinnati (2002), cada vez que se utiliza un terminal, un registro de uso se almacena en el terminal en la forma de un mensaje. Estos mensajes pueden registrar la inscripción de una persona, una anulación de supervisor, o cualquier otra actividad en modo de comandos en la terminal. Cuando se examina un terminal, todos los mensajes se envían al PC Host. Una vez que los mensajes han sido enviados a PC Host, se eliminan desde el lector de mano. En el PC Host, estos mensajes se anexan a los archivos YYYYMMDD.LOG y en su "YYYY" representa el año, "MM" representa el mes y "DD" representa el día. Se crea el archivo YYYYMMDD.LOG por cada día que el software HP-32 se ejecuta y sólo contiene mensajes creados ese día. Una lista de todos los demás mensajes de terminal se almacena en el archivo SYS.LOG y una lista de los errores del sistema se encuentra en el archivo ERR.log. Estos archivos están en formato ASCII simple, con un registro por línea. Estos archivos se pueden ver en la ventana de texto usando el menú Ver o utilizando cualquier editor de texto.

Las matrículas de usuario se manejan de manera diferente a los mensajes de registro. Una inscripción contiene el número de identificación del empleado y su plantilla de la mano de inscripción. Amano Cincinnati (2002), asevera

que cuando se recibe un mensaje de inscripción en el PC Host, el número de identificación y la plantilla de la mano se copian en la base de datos de usuario, USUARIO.DAT. A continuación, se comprueba para un número de identificación correspondiente. Si no se encuentra ningún número de adaptación, la inscripción se almacena en la base de datos de usuario, pero no se transmite a cualquier otro lector en la red. Permanecerá en el terminal hasta que se elimine. Si se encuentra un número de adaptación, la plantilla se coloca en la base de datos. El HP-32 determina entonces qué, sitios deben ser descargados, esto se define en el cuadro de diálogo Grupos y se almacena en el archivo de GROUP.TXT. Las plantillas de mano se descargan a los sitios o terminales adecuados para que se definan en los Sitios y Relojes que se almacenan en los archivos SITE.TXT y CLOCK.TXT. Esto permite a los usuarios acceder a cualquier terminal que está conectado en red con el PC Host.

1.6.1.5.1 HandPunch Software: Sitios

- Un sitio es utilizado por HP-32 para las comunicaciones entre el PC Host y las terminales.

Hasta 31 terminales se pueden incluir en cada sitio (véase Ilustración 1.7).

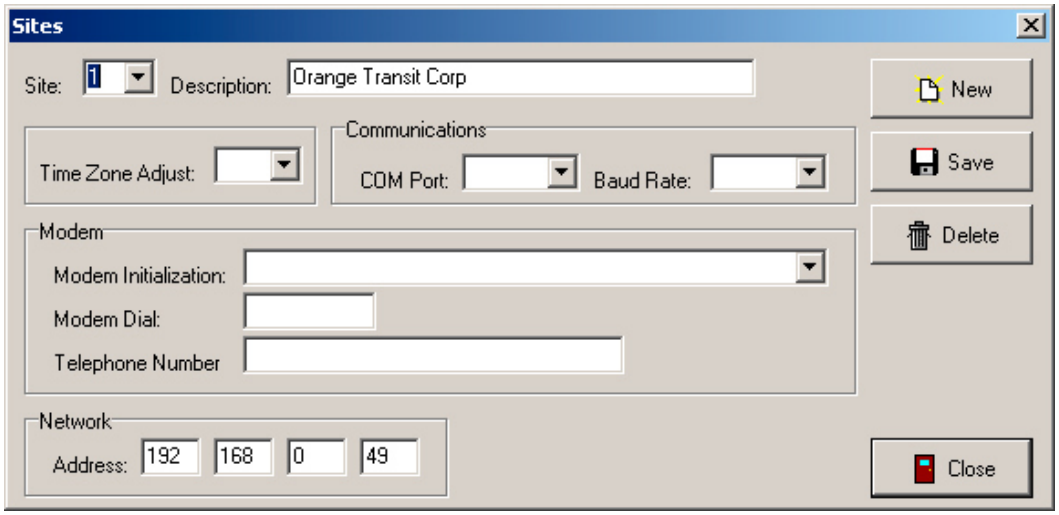


Ilustración 1.7: Creación de un Sitio.

Fuente: Amano Cincinnati, 2002

1.6.1.5.2 HandPunch Software: Relojes

Los parámetros de configuración de cada terminal en un sitio se configuran en el cuadro de diálogo de relojes (véase Ilustración 1.8)

Una vez guardado, esta información se almacena en la definición del reloj (CLOCK.TXT) y configuración (HP.CFG).

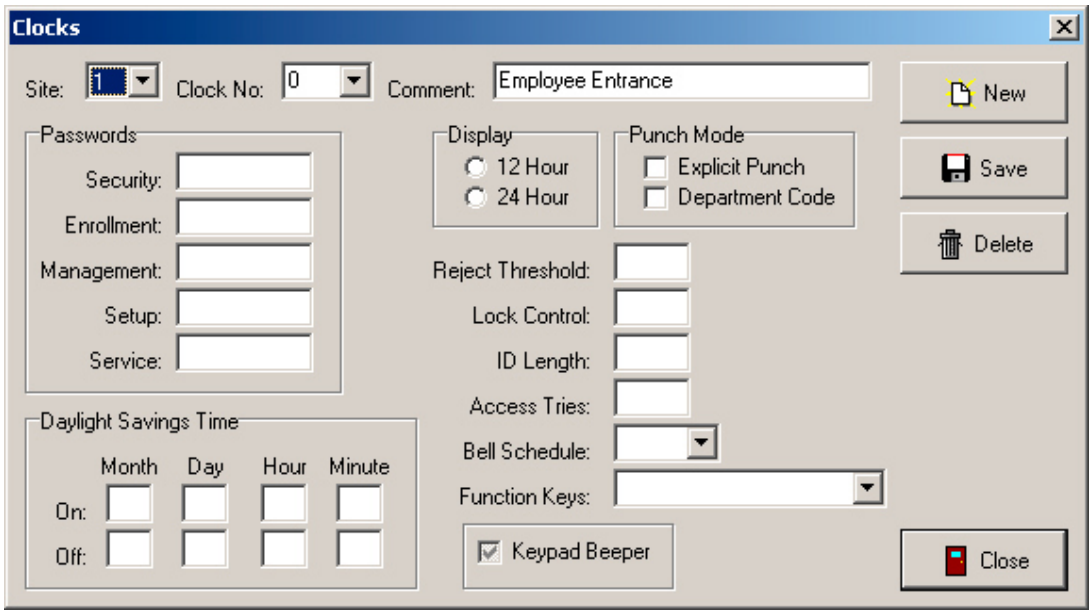


Ilustración 1.8: Mantenimiento de Relojes.

Fuente: Amano Cincinnati, 2002

1.6.1.5.3 HandPunch Software: Encuestar [Poll]

El comando Poll se utiliza para cargar datos de las transacciones y las plantillas de mano biométricos de los terminales seleccionados al PC Host (véase Ilustración 1.9). Cuando en un sitio se realiza un sondeo, se recuperan todas las transacciones registradas desde la última vez recogidas, las transacciones se adjunta al archivo YYYYMMDD.LOG. Se crea un nuevo archivo YYYYMMDD.LOG para cada día. Si inscribieron a cualquier usuario registrado, sus datos de la plantilla de la mano

se recupera y se escribe en el archivo USER.DAT. Además, todas las plantillas de mano para los usuarios que han registrado se recuperan y sus registros asociados en el archivo USER.DAT se actualizan con las últimas plantillas.

Para ejecutar directamente desde el símbolo del DOS del PC Host usar: P[sss[rr]]. Donde sss=Número del Sitio y rr=Reloj o Número del Terminal. Ejemplo: HPUNCH32.EXE P00101 T

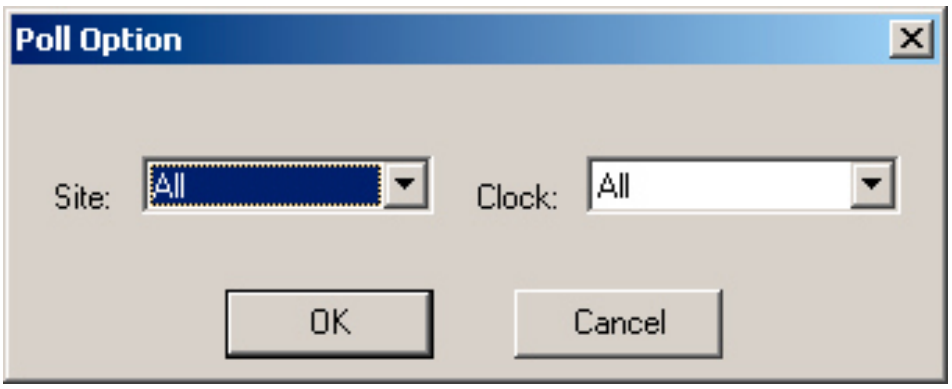


Ilustración 1.9: Comando Encuesta [Poll].

Fuente: Amano Cincinnati, 2002

1.6.1.5.4 HandPunch Software: Restaurar Base de Datos [Restore Database]

Este comando se utiliza para restaurar la base de datos de plantillas de manos a un terminal (véase Ilustración

1.10). Esto se hace en la limpieza de la base de datos primero en el terminal(s) seleccionado y luego la descarga de plantillas de mano previamente asignados desde el archivo USER.DAT desde el PC Anfitrión.

Para ejecutar directamente desde el símbolo del DOS del PC Host usar: R[sss[rr]]. Donde sss=Número del Sitio y rr=Reloj o Número del Terminal. Ejemplo: HPUNCH32.EXE R00101 T

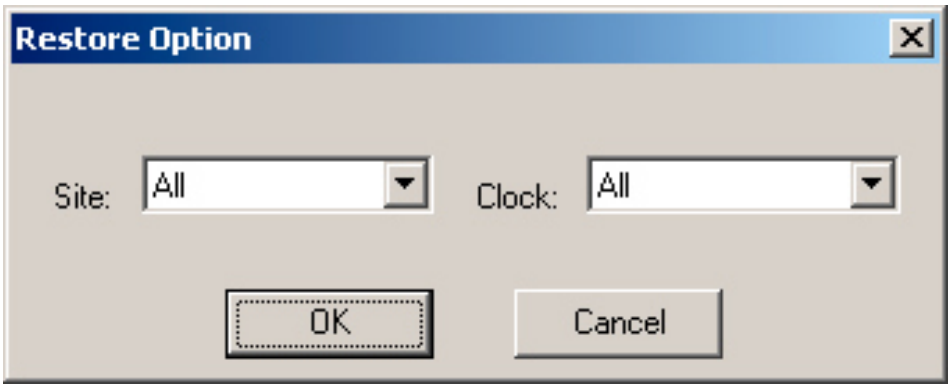


Ilustración 1.10: Comando Encuesta [Poll].

Fuente: Amano Cincinnati, 2002

1.6.1.5.5 HandPunch Software: Obtener Base de Datos [Get Database]

Este comando se utiliza para recuperar las plantillas de la mano de los sitios o terminales seleccionados para todos los usuarios que sólo están inscritos en el fichero

USER.txt en el PC Host (véase Ilustración 1.11). Esto permite que el archivo USER.DAT se actualice con las plantillas de mano más recientes.

Para ejecutar directamente desde el símbolo del DOS del PC Host usar: G[sss[rr]]. Donde sss=Número del Sitio y rr=Reloj o Número del Terminal. Ejemplo: HPUNCH32.EXE G00101 T

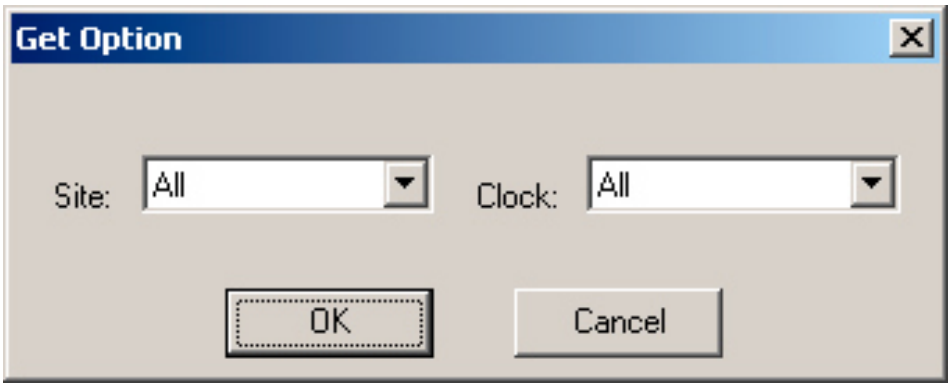


Ilustración 1.11: Comando Obtener Base de Datos [Get Database].

Fuente: Amano Cincinnati, 2002

1.6.1.5.6 HandPunch Software: Obtener Base de Datos Completa [Get Full Database]

Este comando se utiliza para recuperar las plantillas de la mano de los Sitios o terminales para todos los usuarios seleccionados independientemente si están

inscritos en el expediente USER.txt en el PC Host (véase Ilustración 1.12).

Para ejecutar directamente desde el símbolo del DOS del PC Host usar: F[sss[rr]]. Donde sss=Número del Sitio y rr=Reloj o Número del Terminal. Ejemplo: HPUNCH32.EXE F00101 T

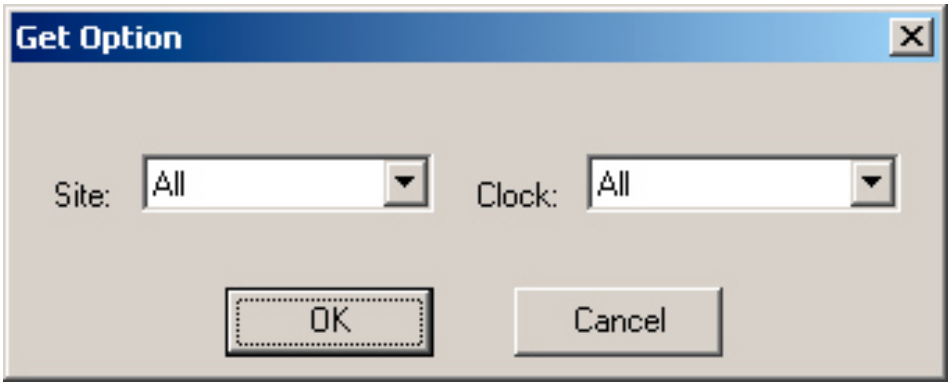


Ilustración 1.12: Comando Obtener Base de Datos Completa [Get Full Database].

Fuente: Amano Cincinnati, 2002

1.6.1.5.7 HandPunch Software: Enviar Hora [Send Time]

Este comando envía la fecha y la hora del PC Host a los Sitios o terminales seleccionados (véase Ilustración 1.13)

Para ejecutar directamente desde el símbolo del DOS del PC Host usar: T[sss[rr]]. Donde sss=Número del Sitio y rr=Reloj o Número del Terminal. Ejemplo: HPUNCH32.EXE T00101 T

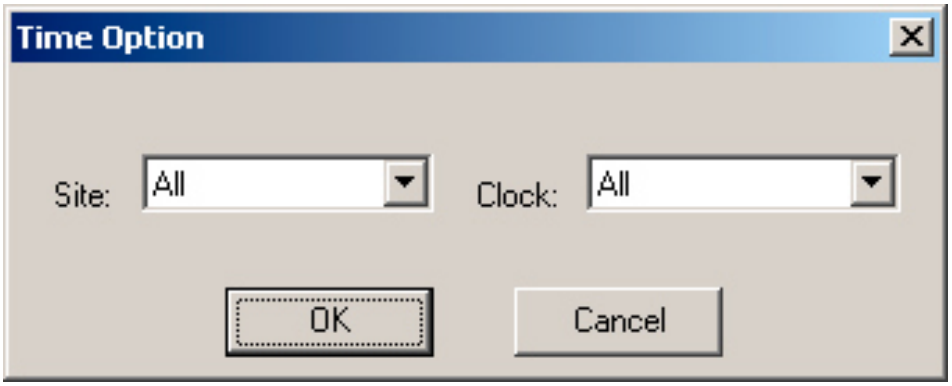


Ilustración 1.13: Comando Enviar Hora [Send Time].

Fuente: Amano Cincinnati, 2002

1.6.1.5.8 HandPunch Software: Enviar Configuración [Send Config]

Este comando descarga datos de configuración almacenados en el archivo de HP.CFG a los terminales especificados (véase Ilustración 1.14). Los datos almacenados en el archivo HP.CFC se introduce en el reloj, campanas, zona horaria, y los cuadros de diálogo de vacaciones.

Para ejecutar directamente desde el símbolo del DOS del PC Host usar: C[sss[rr]]. Donde sss=Número del Sitio y

rr=Reloj o Número del Terminal. Ejemplo: HPUNCH32.EXE C00101 T

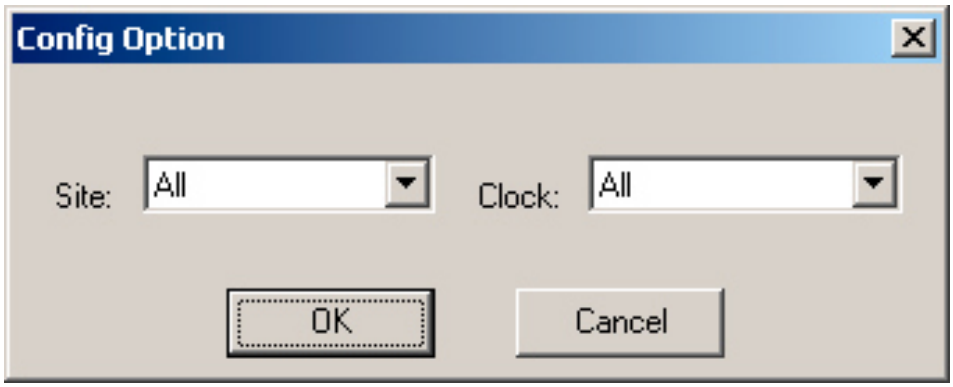


Ilustración 1.14: Comando Enviar Configuración [Send Config].

Fuente: Amano Cincinnati, 2002

1.6.2 SquareNet

SquareNet Gestión Humana es un software que se compone de Nómina, Control de Asistencias, Recursos Humanos y Control de Visitas, cuatro módulos que permiten mantener completo control y manejo sobre la asistencia, contratos, acciones de personal y pago de sueldos de sus empleados de una manera fácil, rápida y eficaz (véase Ilustración 1.15).

> Trabaja en red, soporta entornos multi-empresa, multi-usuario y multi-sucursal, tiene un completo control de seguridad por perfiles de usuario y es totalmente parametrizable.
>
> Cuenta además, con una gran variedad de nuevas herramientas visuales, como: una navegación en los datos más intuitiva, graficación de datos, exportación directa a Excel, envío de mail, perfiles de usuario, auditoria y completo control sobre todas sus operaciones. Para usuarios expertos herramientas de administración, como definición de mensajes de error, actualizaciones automáticas y un editor de sentencias SQL directamente a la base de datos. (SquareNet Software, 2008)

El software de SquareNet es actualmente usado para la gestión del recurso humano de la PUCESA y a la ves de gran importancia para el desarrollo de este proyecto ya que en su base de datos aloja horarios e información personal y administrativa de los trabajadores.

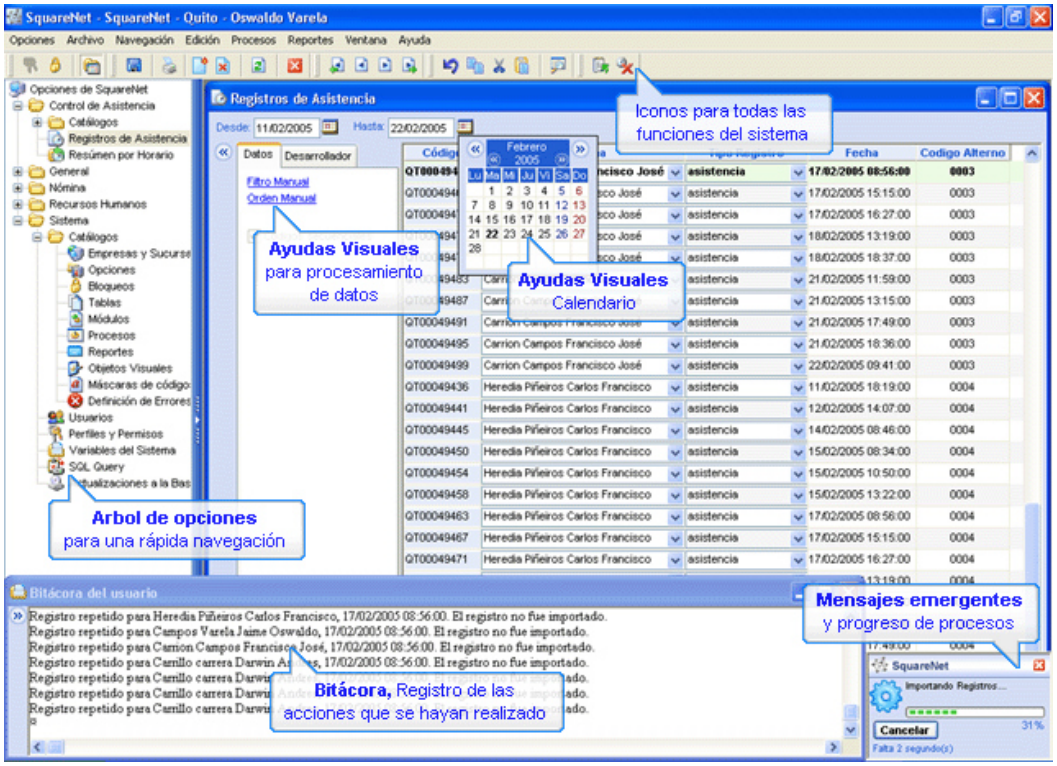


Ilustración 1.15: SquareNet Software.

Fuente: SquareNet Software, 2008

1.6.2.1 Módulo de Control de Asistencia

Este módulo administra toda la información relacionada con el cumplimiento y la asistencia del personal. Tiene máxima flexibilidad y orientación industrial para cumplir hasta con el más complejo esquema de horarios y turnos, incluso con soporte para fórmulas y miniscripts personalizables al estilo de Microsoft Excel para cálculos definidos por el usuario.

> Se puede definir los horarios de trabajo, por grupos, por persona, o incluso diario por persona, permitiendo especificar también su puesto de trabajo o centro de costo diario lo que facilita increíblemente el costeo de los empleados en escenarios complejos.
>
> Tiene conectividad directa con lectores biométricos de SquareNet a través del software SquareBridge, u otros modelos como HandPunch. (SquareNet Software, 2008)

El módulo de Control de Asistencia perteneciente al software de SquareNet es el más usado por la Dirección de Talento Humano para contrastar horarios y registros realizados en los dispositivos biométricos por parte de los trabajadores de la PUCESA, mencionada comparación es base para la elaboración de roles de pago al término del mes.

1.6.3 Ruby y RoR

Según Rajshekhar (2008), Ruby on Rails es un framework de aplicaciones web de código abierto ideal para la construcción de aplicaciones de negocio, acelerar y simplificar la creación de sitios web con bases de datos. Ha sido desarrollado en la plataforma Ruby. Desarrolla aplicaciones web con bases de datos de acuerdo con el patrón Model-View-Controller.

El mejor marco para el desarrollo de aplicaciones web es el que mejora la productividad. Rajshekhar (2008), concluye que un marco que reduzca código repetitivo y también reduzca la curva de aprendizaje es el que aumenta la productividad. Hay una gran cantidad de marcos que se adaptan al desarrollo de aplicaciones web. Pero la mayoría de ellos fracasan en uno de los dos puntos que rigen la productividad. O bien el marco reduce el código repetitivo o es más fácil de aprender. El logro de un equilibrio entre los dos se ve como una tarea difícil. Es aquí que Ruby y Ruby on Rails (RoR), posee una puntuación por encima de la mayoría de sus contemporáneos.

Siempre es una buena idea saber acerca de las especificaciones de la herramienta con la que se trabaja, antes de manejar la herramienta. Las herramientas son Ruby y RoR, Ruby como el lenguaje y RoR como marco construido sobre Ruby.

1.6.3.1 Ruby

En 1995, Yukihiro Matsumoto lanzó la primera versión de Ruby. La principal razón dada por el Sr. Matsumoto según Rajshekhar (2008), era que quería un lenguaje de "scripting" que optimizara la manera en que un programador desarrollara software. Esto significa que las características de Ruby son tales que optimizan la forma en que el software se desarrolla. ¿Cuáles son estas características?

- **Interpretada:** Ruby es un lenguaje interpretado. Por lo tanto, cada vez que se realiza un cambio en el código fuente, no es necesario compilar el código y ejecutarlo para ver el efecto del cambio. Como resultado de esta función, el ciclo de compilación de código de gestión se convierte en el ciclo de gestión de código.

- **Puramente Orientada a Objetos:** está orientado a objetos, eso significa que todo en Ruby es un objeto que incluye primitivos tipos de datos y números. Además, es compatible con todas las características requeridas por un lenguaje orientado a objetos.

- **Funcional:** Ruby apoya la programación funcional utilizando bloques.

- **Duck Typing:** También se conoce como dinámica de Mecanografía. Ruby decide sobre el tipo de variable, mientras que el programa se está ejecutando. En otras palabras, si un objeto se parece a un pato, suena como un pato, entonces es un pato.

- **Administración de Memoria Automática:** el también conocido Garbage Collection. Como en cualquier lenguaje de muy alto nivel (VHLL), Ruby ofrece Recolección de Basura fuera de la caja, por lo tanto no es preocupante las pérdidas de memoria física.

- **Threading:** La versión estable actual de Ruby ofrece "casi" plataforma independiente de roscado con hilos verdes (hilos utilizados en el nivel de espacio de usuario). "Casi" porque los hilos de Ruby se simulan

en la máquina virtual en lugar de correr hebras de SO como nativas.

- **Reflexión:** Ruby ofrece un programa con la capacidad de "mirar a sí mismo" mientras se ejecuta. Esta capacidad es conocido por diferentes términos, tales como la reflexión, la introspección, y así sucesivamente. El uso de la reflexión en un programa puede modificar ciertos aspectos de sí mismo durante la ejecución, o crear un nuevo objeto en tiempo de ejecución en base a los requerimientos de la época.

1.6.3.1.1 Jruby

JRuby es un intérprete de Ruby escrito por un programador llamado Jan Arne Petersen en 2001. Según Edelson y Liu (2008), tras la liberación de Ruby 1.8 en 2003 y luego la liberación del framework web Ruby on Rails en el 2004, una cantidad significativa de esfuerzo se ha puesto en desarrollar JRuby, especialmente en las áreas de compatibilidad y rendimiento. En septiembre de 2006, Sun Microsystems aprobó efectivamente JRuby cuando contrató a dos desarrolladores de la iniciativa, Charles Nutter y Thomas Enebo, para trabajar en JRuby a tiempo completo.

Desde entonces, un tercer desarrollador principal, Nick Sieger, se ha convertido en un empleado de Sun.

Para Sun, JRuby representa una oportunidad para ampliar la prevalencia de la máquina virtual de Java. Aunque la JVM fue originalmente ligada al lenguaje Java, la aparición de proyectos como JRuby, Jython (Implementación en Java de Python), han demostrado que la JVM puede alojar una amplia variedad de lenguajes.

1.6.3.2 RoR (Ruby on Rails)

RoR es un operador reciente en el mundo de los entornos de aplicaciones web a la altura de J2EE/JEE por sus funcionalidades. El otro aspecto que rige la aceptación de un marco de trabajo es su filosofía. Por lo tanto, es necesario mirar tanto los aspectos de funcionalidad, como también de su filosofía misma que es válida para todas las versiones de RoR.

1.6.3.2.1 Filosofía

Según Rajshekhar (2008), la filosofía de RoR depende de dos principios, el primero:

- Don’t Repeat Yourself (DRY): No te repitas, si se aplica correctamente, reduce la duplicación de tareas las que llevan a la dificultad en la modificación, mantenimiento y finalmente a la inconsistencia en un proyecto. En RoR, se puede ver este principio en casi todo, desde los componentes reutilizables en forma de plug-ins hasta la forma que se asignan las tablas de las bases de datos.

Y el segundo según Rajshekhar (2008):

- Convention-over-Configuration (CoC): La configuración se ha hecho cargo de los entornos de aplicaciones Web tanto que incluso una tarea simple, como la aplicación de la validación "campo obligatorio" para un solo campo requiere entradas en un archivo XML. En RoR, el principio es suministrar información acerca de sólo aquellos aspectos que son diferentes de los ajustes normales de aplicación. El

ORM (Object Relational Mapping) marco proporcionado por RoR es un ejemplo del principio de la Convención sobre-Configuración. A menos que especifique otro nombre para un objeto ORM, el objeto se utiliza con el nombre de la tabla a la que está asignada. Mientras que en el caso de los marcos ORM basado en la configuración, como Hibernate, el mapeo de cada tabla junto con sus columnas tiene que ser dada en el archivo de configuración. Así, un cambio en el esquema significa un cambio en el archivo de configuración. Sin embargo, en el caso de RoR, un cambio en el esquema no significa un cambio en el objeto a menos que el nombre de la tabla en sí cambie.

1.6.3.2.2 Características

Las características basadas en la filosofía “DRY” y principios “Convención sobre configuración” son los que hacen de RoR tan atractivo para el desarrollo de sitios web dinámicos.

1.6.3.2.3 Conceptos y Componentes

RoR es un framework basado en Ruby que implementa el patrón MVC.

Hay dos puntos clave en esta definición:

- Se trata de un marco basado en Ruby.
- Implementa el patrón MVC

RoR es un marco basado en Ruby

Rajshekhar (2008), afirma que la naturaleza dinámica y abierta de Ruby hace que sea una opción atractiva para construir marcos. Dada la facilidad de meta-programación, reflexión, bloques e iteradores, junto con el manejo de excepciones, tienen un lenguaje que pudiera atender cualquier nivel de una aplicación web.

Es evidente cómo Ruby facilita la meta-programación a partir de Active Record, el marco ORM dentro de RoR. Basado en el nombre de la clase, RoR (básicamente Ruby

construye) lee el esquema y crea los objetos de la clase sobre la base de los datos recuperados de la tabla en la marcha. Un método de acción se comunica con la vista correspondiente utilizando una variable de instancia y no a través de los parámetros de uso explícitos enviados a través de objetos de petición o de objetos de sesión. Aparte de estos, RoR hace un uso intensivo estructuras y bloques de código anónimos para reducir la configuración.

RoR implementa el patrón MVC

Para Rajshekhar (2008), MVC es un patrón de diseño que ofrece una demarcación clara entre los tres aspectos de una lógica de acceso a datos de la aplicación, lógica de flujo de control, y lógica de presentación. Estos tres aspectos han sido considerados como:

- Modelo.- Representa los datos procesados por la aplicación. Proporciona un enlace al almacenamiento persistente (almacén de datos).

- Vista- la lógica que corresponde a la visualización de los datos. Es el único aspecto de MVC que interactúa directamente con el usuario .

- Controlador.- Representa la lógica de flujo de control. Las decisiones sobre qué punto de vista ha de ser llamado para visualizar los datos actuales y qué parte del modelo se tiene que actualizar. Se encuentra en el límite de su aplicación e intercepta cada petición. Luego llama al modelo correspondiente para actualizar o recuperar datos, y luego escoge la vista apropiada para mostrar los datos.

Rajshekhar (2008), asevera que RoR implementa MVC proporcionando tres capas o componentes como parte del marco. Ellos son: Active Record, Action View y Action Controller

Action Controller y Action View juntos se conocen como Action Pack.

Active Record

Active Record es el “modelo” en RoR. Los datos almacenan los componentes del modelo y proporciona funcionalidad para trabajar con los datos. Aparte de ser el componente de modelo, Active Record es también un marco ORM.

Action View

Abarca la lógica para la presentación de los datos del componente de modelo. Action View es el componente Vista de RoR. Las funcionalidades proporcionadas por Action View abarcan desde la creación de plantillas para “Ajaxifying” la página web.

Action Controller

El controlador orquesta el flujo de la lógica. En una aplicación web, es el controlador el que regula y organiza el flujo de la lógica de la aplicación. El controlador se encuentra en el límite de la solicitud e intercepta todas las peticiones, sobre la base de éstas, se actualiza el objeto de modelo correspondiente y llama

a la lógica de la vista para mostrar los datos actualizados. En RoR, el controlador de acción proporciona las funcionalidades del controlador.

1.6.4 HTML5

Aunque similares, cada versión de HTML utiliza etiquetas diferentes, y ya que en sí mismo el HTML5 tiene objetivos más ambiciosos de los que pretendían cubrir versiones anteriores, éste tiene su propio y más extenso juego de etiquetas.

HTML4 tiene múltiples ventajas como: ampliamente aceptado hoy en día, su implementación está prácticamente en todos los navegadores, es permisivo con pequeños errores de código (esto puede verse también como desventaja), si la compatibilidad entre navegadores es un requerimiento primordial puede ser la mejor opción.

Por otro lado algunas desventajas de HTML4 que Herrera (2011) menciona, que pese a las funcionalidades propietarias añadidas a lo largo de los años han creado cierta flexibilidad, no son comparables con las de HTML5.

Las capacidades para formularios son pocas, limitadas y esencialmente feas. La mayoría de los navegadores soportan alguna forma de JavaScript, lo que permite mayor dinamismo, pero las implementaciones siguen teniendo amplias diferencias en algunos casos.

Así mismo Herrera (2011) afirma que el XHTML es un estándar estricto basado en el HTML4 y XML (por lo que su implementación es también casi universal), no es permisivo con los errores, como sí lo es HTML4, lo que obliga a formar correctamente un documento HTML como sucede con un documento XML.

W3C trabajaba en XHTML2, esto no continuará más. Se ha trabajado en la serialización de HTML5 conocida como XHTML5 por ahora las únicas diferencias son el tipo MIME que utiliza y que la etiqueta DOCTYPE es opcional.

HTML5 tiene ventajas como poseer una gran cantidad de funcionalidades que los desarrolladores sólo lograban con el uso de algún plug-in de terceros como applets de java o flash embebidos en el código. Las mejoras en el manejo de multimedia son ampliamente superiores; imágenes, video, audio y fuentes de texto son mucho más

manipulables. En muchos casos, estas mejoras se pueden tomar como una respuesta directa a Flash. Finalmente entre las desventajas de HTML5 se puede afirmar que es tan nuevo que algunas personas que todavía usan computadoras viejas podrían usar navegadores que no pueden visualizarlo aún.

En resumen, HTML5 proporciona una manera de hacer un código más limpio, más fácil de leer y escribir, cubriendo al mismo tiempo y de mejor manera cada vez una mayor demanda de funcionalidades por parte de programadores, diseñadores y usuarios.

1.6.5 JBoss

JBoss Application Server es una plataforma certificada Java EE para el desarrollo y despliegue de aplicaciones Java Enterprise. JBoss Application Server proporciona la gama completa de características Java EE, así como los servicios de empresa ampliada, incluyendo “clustering”, “caching” y “persistence”.

Según Marchioni (2013), JBoss cumple con altos estándares de confiabilidad, eficiencia, solidez, y se utiliza para construir potentes y seguras aplicaciones Java EE. Es compatible con las zonas más importantes de la programación de Java Enterprise incluyendo EJB, contextos, inyección de dependencias, JAX-WS y servicios web JAX-RS, el marco de seguridad, y más.

La séptima versión de JBoss AS es muy diferente de todas las demás versiones de servidor y como una cuestión de hecho ha mejorado en varios puntos clave.

1.6.5.1 TorqueBox

Marchioni (2013), afirma que TorqueBox proporciona un entorno de nivel empresarial que no sólo brinda una completa compatibilidad con Ruby-on-Rails, también va más allá de la funcionalidad ofrecida en ambientes tradicionales Rails.

En lugar de construir una aplicación con plataforma Ruby desde 0, TorqueBox aprovecha la funcionalidad existente de JBoss.

TorqueBox es un producto de la Comunidad de JBoss, y es completamente software de código abierto. TorqueBox está licenciado bajo la LGPL.

CAPÍTULO II
METODOLOGÍA

2.1 Metodología de Desarrollo

"Extreme Programming (XP) es una manera ligera, eficiente, bajo riesgo, flexible, predecible, científico y divertido para desarrollar software." (Beck K, 2000)

> La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código. Los objetivos de XP son muy simples: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuándo lo necesita. El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software. (Beck K., 2000)

XP se distingue de otras metodologías de desarrollo ágil de software por:

- Retroalimentación rápida, concreta, y siguiendo los ciclos cortos.

- Enfoque de planificación gradual con un rápido plan general de que se espera evolucionar a través de la vida del proyecto.

- Capacidad de programar de forma flexible la funcionalidad de la aplicación, en respuesta a las cambiantes necesidades del negocio.

- Dependencia de las pruebas automatizadas escritas por los programadores y clientes para monitorear el progreso del desarrollo y permitir que el sistema evolucione, para detectar los defectos antes de tiempo.

- Dependencia de la comunicación oral, pruebas, y código fuente para comunicar estructura e intención al sistema.

- Dependencia de un proceso de diseño evolutivo que dura mientras el sistema permanece.

- Dependencia de la estrecha colaboración de los programadores con conocimientos ordinarios.

- Dependencia de las prácticas que funcionan tanto con los instintos de corto plazo de los programadores y los intereses a largo plazo del proyecto.

2.1.1 Fases de Extreme Programming

A continuación la Ilustración 2.1 muestra las fases de Extreme Programming con las que entrega software funcional basado en la interpretación de Pressman (2010), se adjuntan los documentos base para el desarrollo de aplicaciones. Luego durante todo el capítulo se detallan las reglas más importantes que identifican a XP.

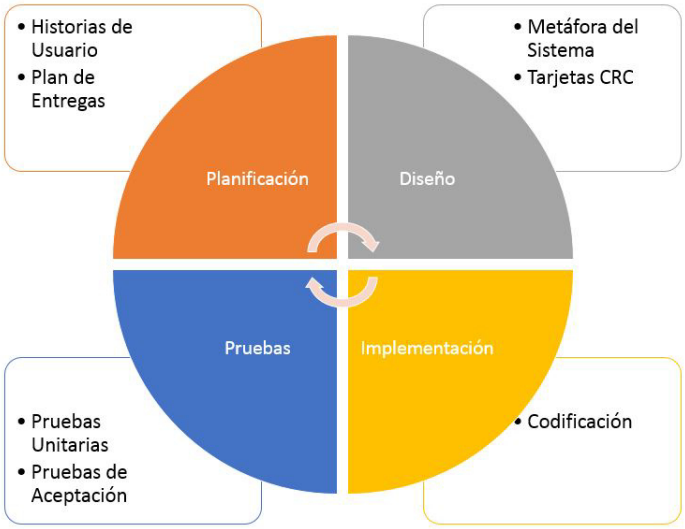


Ilustración 2.1: Fases de Extreme Programming.

Elaborado: Altamirano, 2014

2.1.1.1 Planificación

2.1.1.1.1 Historias de Usuarios

Las historias de usuario tienen el mismo propósito que los casos de uso, pero no son lo mismo. Se utilizan para crear las estimaciones de tiempo para las reuniones de planificación. También se utilizan para documentar necesidades de gran tamaño. Las historias de usuario son escritas por los usuarios como por ejemplo las cosas que el sistema necesita para hacer por ellos. Son similares a

los casos de uso, excepto que no se limitan a la descripción de una interfaz de usuario. El texto es escrito por el usuario sin tecno-sintaxis.

Según Beck y Fowler (2001), las historias de usuario también impulsan la creación de las pruebas de aceptación. Una o varias pruebas de aceptación deben ser creadas para verificar que la historia de usuario se ha implementado correctamente.

Uno de los mayores malentendidos con las historias de usuario es la forma en que difieren de las especificaciones de requisitos tradicionales. La mayor diferencia está en el nivel de detalle. Para Beck y Fowler (2001), las historias de los usuarios sólo deben proporcionar suficientes detalles para hacer una estimación razonable de cuánto tiempo la historia se llevará a la práctica. Cuando llegue el momento de poner en práctica la historia, los desarrolladores irán al usuario y pedirán recibir una descripción detallada de los requisitos cara a cara.

Los desarrolladores estiman cuánto tiempo podría tomar implementar las historias de usuario. Cada historia

tendrá una 1, 2 o 3 semanas en "tiempo de desarrollo ideal". Este tiempo de desarrollo ideal es cuánto tiempo se tarda en poner en práctica la historia en el código, si no hay distracciones, otras tareas y conociendo exactamente qué hacer.

Otra diferencia entre las historias y un documento de requisitos es el enfoque en las necesidades del usuario. Se debe tratar de evitar los detalles de tecnología específica, el diseño de base de datos y algoritmos, se trata de mantener historias centradas en las necesidades del usuario.

2.1.1.1.2 Reunión de Planificación de Entregas

Una reunión de planificación de entregas se utiliza para crear un plan de entregas. El plan de entregas se utiliza para crear planes de iteraciones individuales.

Beck y Fowler (2001), afirma que esta reunión es importante para los técnicos ya que pueden tomar decisiones técnicas, como así también a quien competa tomar decisiones de negocio. La planificación de entregas

tiene un conjunto de reglas que permite a todos los involucrados con el proyecto tomar sus propias decisiones. Las reglas definen un método para negociar un calendario con el que todo el mundo puede comprometerse.

La esencia de la reunión de planificación de entregas es para estimar cada historia de usuario en términos de semanas de programación ideales. El usuario decide qué historia es la más importante o tiene la más alta prioridad para ser completada.

Las historias de usuario se imprimen o se escriben. Según Beck y Fowler (2001), la velocidad de proyecto puede ser calculado determinando cuántas historias se pueden implementar en un plazo determinado (tiempo) o el tiempo que un conjunto de historias se tardará en terminar (ámbito de aplicación).

Las iteraciones individuales planificadas en detalle comienzan justo antes de cada iteración y no por adelantado.

Una vez creado el plan de entregas final es tentador cambiar las estimaciones para las historias de usuario. No se debe hacer esto. Las estimaciones son válidas y serán requeridas como está durante las reuniones de planificación de la iteración. Subestimar causará problemas en el futuro. Beck y Fowler (2001), recomienda que en lugar de negociar un plan de entregas aceptable es preferible negociar hasta que los desarrolladores, usuarios y gerentes estén de acuerdo con el plan de entregas.

2.1.1.1.3 Plan de Entregas

Después de que las historias de usuario se han escrito se debe ejecutar una reunión de planificación de entregas para crear un plan de entregas. El plan de entregas especifica qué historias de usuario se va a implementar para cada versión del sistema y el tiempo de las entregas. Estas historias seleccionadas se traducen en tareas de programación individuales que se ejecutarán durante la iteración para completar las historias.

Para Beck y Fowler (2001), las historias también se traducen en las pruebas de aceptación durante la

iteración y las iteraciones posteriores para comprobar si las historias se terminan correctamente y siguen funcionando.

2.1.1.1.4 Desarrollo Iterativo

El desarrollo iterativo agrega agilidad al proceso de desarrollo. Divide el calendario de desarrollo en cerca de una docena de repeticiones de 1 a 3 semanas de duración. Una semana es la mejor opción a pesar de que parezca muy corto. Beck y Fowler (2001), aseveran que es esencial mantener la longitud de iteración constante durante todo el proyecto. Este es el latido del corazón del proyecto. Es, esta constante la que hace la medición del progreso y la planificación sencilla y fiable en XP.

No programar las tareas de programación con antelación. En su lugar realizar una planificación al comienzo de la iteración de lo que se hará. La planificación “Just in Time” es una manera fácil de estar al tanto de los requerimientos cambiantes de los usuarios.

También es contrario a las normas el mirar hacia adelante y tratar de poner en práctica, todo lo que no está previsto para esta iteración. Habrá mucho tiempo para implementar esa funcionalidad cuando se convierta en la historia más importante en el plan de entregas.

Beck y Fowler (2001), recomiendan tomar los plazos de iteración en serio. Seguir el progreso durante una iteración. Si parece no terminar todas las tareas lo recomendable es, llamar a otra reunión de planificación de la iteración para re-estimación, y eliminar algunas de las tareas.

Concentrar el esfuerzo en completar las tareas más importantes elegidas por el usuario, en lugar de tener varias tareas sin terminar elegidas por los desarrolladores.

2.1.1.1.5 Planificación de la Iteración

Una reunión de planificación de la iteración, se llama al comienzo de cada iteración para producir el plan de tareas de programación de esa iteración. Cada iteración

es de 1 a 3 semanas de duración. Las historias de usuario son elegidas para esta iteración por el usuario desde el plan de lanzamiento con el fin de seleccionar primero las más valiosas para el usuario.

Las historias de usuario y las pruebas fallidas se descomponen en las tareas de programación en las que se apoyarán. Las tareas están escritas en tarjetas, como las historias de usuario. Si bien las historias de usuario se encuentran en el idioma del usuario, las tareas están en el idioma del programador. Las tareas duplicadas se pueden eliminar. Estas tarjetas de trabajo serán el plan detallado para la iteración.

Beck y Fowler (2001), afirman que es importante para el desarrollador que acepta una tarea sea él quien estime cuánto tiempo se tardará en terminar. Las personas no son intercambiables y la persona que se va a hacer cargo de la tarea debe estimar cuánto tiempo tomará.

Cada tarea debe ser estimado como 1, 2, o 3 (1/2 si es necesario) días ideales en la duración de la programación. Los días de programación ideales son cuánto tiempo le tomaría para completar la tarea si no hay

distracciones. Las tareas cortas que son de 1 día se agruparán. Las tareas que son más de 3 días se desglosarán.

Ahora la velocidad de proyecto se utiliza de nuevo para determinar si la iteración es archivada o no. Beck y Fowler (2001), sostiene que el total de las estimaciones de tiempo en días ideales de programación no debe ser superior a la velocidad del proyecto de la iteración anterior. Si la iteración tiene demasiadas historias, entonces el usuario debe elegir las que se deje para una iteración posterior.

Si la iteración tiene muy pocas historias, otra puede ser aceptada. La velocidad en el día de trabajo (planificación de la iteración) anula la velocidad en semanas de la historia (planificación de entrega), ya que es más preciso.

Es importante considerar la importancia de las pruebas unitarias y refactorización. Una deuda en cualquiera de estas áreas produce pérdida de tiempo. Evitar en lo posible el agregar cualquier funcionalidad antes de que

sea programada. Simplemente agregar lo que se necesita para hoy.

El proceso de planificación se basa en la fría realidad de estimaciones consistentes, esquivar crea más problemas que soluciones.

Beck y Fowler (2001), confirman que puede ser necesario volver a estimar todas las historias y volver a negociar el plan de lanzamiento de tres a cinco iteraciones, esto es normal. Siempre y cuando se haya puesto en práctica las historias más valiosas primero y que siempre se haga todo lo posible para sus usuarios y la gestión.

Un estilo de desarrollo iterativo puede añadir agilidad al proceso de desarrollo.

2.1.1.2 Diseño

2.1.1.2.1 La Simplicidad es la Clave

Un diseño simple siempre toma menos tiempo para terminar una cuestión compleja. Si se encuentra algo que es complejo, reemplazarlo con algo simple. Siempre es más rápido y más barato reemplazar pronto el código complejo, antes de perder mucho más tiempo en él.

Dentro del proyecto, el equipo decide lo que es simple. Beck (2000), recomienda cuatro cualidades subjetivas; comprobable, navegable, comprensible y explicable.

Comprobable significa que puede escribir pruebas unitarias y pruebas de aceptación para comprobar automáticamente si hay problemas. Esto afecta el diseño general y el acoplamiento de los objetos de la aplicación.

Navegable es la cualidad de ser capaz de encontrar lo que desea cuando lo desee. Los buenos nombres le ayudan a

encontrar las cosas. La utilización de polimorfismo, delegación y la herencia ayuda a encontrar las cosas cuando es necesario.

Comprensible es obvio, pero muy subjetivo. Un equipo que ha estado trabajando con un sistema por mucho tiempo lo entenderá. Así que explicable es también una cualidad que significa fácil de mostrar a la gente nueva cómo funciona todo.

Los diseños simples a menudo vienen después de que el proyecto ha estado funcionando durante un tiempo. El mejor enfoque es crear código sólo por las funciones que se están implementando mientras se busca el conocimiento suficiente para revelar el diseño más simple.

Hay que mantener las cosas tan simples como sea posible el mayor tiempo posible y nunca agregar funcionalidad antes de que se programe.

2.1.1.2.2 Metáfora del Sistema

La metáfora del sistema es en sí un diseño simple con ciertas cualidades. Según Beck (2000), la cualidad más importante es ser capaz de explicar el diseño del sistema para las personas nuevas sin recurrir a enormes documentos. Un diseño debe tener una estructura que ayude a las personas nuevas a comenzar contribuyendo rápidamente. La segunda cualidad es un diseño que nombra las clases y métodos consistentemente.

Lo que designa el nombre de los objetos es muy importante para entender el diseño global del sistema y la reutilización de código. Ser capaz de adivinar lo que podría ser nombrado si ya existía y estar en lo correcto la mayor parte del tiempo es un gran ahorro de tiempo.

Se recomienda un sistema que nombre objetos que todo el mundo puede relacionar.

2.1.1.2.3 Tarjetas CRC

Las tarjetas clase, responsabilidades y colaboración (CRC) son usadas para diseñar el sistema en el equipo. El mayor valor de las tarjetas CRC es permitir a la gente romper con el procedimiento del pensamiento y plenamente apreciar la tecnología de objetos. Las tarjetas CRC permiten que equipos enteros de proyectos contribuyan al diseño. Cuantas más personas puedan ayudar a diseñar el sistema mayor será el número de buenas ideas incorporadas.

Smith (2001), confirma que las tarjetas CRC se utilizan para representar objetos. La clase del objeto se puede escribir en la parte superior de la tarjeta, las responsabilidades se enumeran en la parte izquierda, las clases colaboradoras se listan a la derecha de cada responsabilidad. Se dice "se puede escribir" porque en una sesión de CRC puede sólo necesitarse unas cuantas tarjetas con el nombre de clase y no todas escritas en su totalidad.

Según Beck (2000), una sesión de CRC inicia con alguien simulando al sistema hablando de los objetos que enviarán

mensajes a otros objetos. Al avanzar a través del proceso las debilidades y problemas son fácilmente descubiertos. Las alternativas de diseño se pueden explorar de forma rápida mediante la simulación del diseño que se propone.

Una de las mayores críticas a las tarjetas CRC es la falta de diseño por escrito. Esto por lo general no es necesario ya que el diseño con estas tarjetas parece obvio. Si se necesita un registro más permanente, una tarjeta para cada clase puede ser escrito en su totalidad y mantenerse como documentación.

2.1.1.3 Implementación (Codificación)

2.1.1.3.1 Usuario Disponible

Beck (2000), asevera que uno de los pocos requisitos de la programación extrema es que el usuario esté siempre disponible no solo para ayudar al equipo de desarrollo sino para ser parte de él.

Todas las fases de un proyecto XP requieren la comunicación con el usuario que de preferencia sea un experto y no un aprendiz.

Las historias de usuario son escritas por el usuario con ayuda de los desarrolladores para permitir estimaciones de tiempo y prioridad. La mayoría de las funcionalidades deseadas del sistema deben estar cubiertas.

Según Jeffries (2000), durante la reunión de planificación de entrega el usuario tendrá que negociar una selección de historias de usuario que se incluye en cada lanzamiento previsto. Los usuarios deben tomar las decisiones que afectan a sus objetivos de negocio. La reunión de planificación de entrega permite a los usuarios dar a los desarrolladores retroalimentación.

Debido a que los detalles se dejan fuera de las historias de usuario, los desarrolladores tendrán que hablar con los usuarios para obtener la completa información para terminar una tarea.

El usuario ayudará con las pruebas funcionales ya que verificará sus resultados previos a ser lanzado a producción. Calificará las pruebas y permitirá que el sistema continúe en producción o detenerlo.

2.1.1.3.2 Estándares de Codificación.

El código debe tener un formato que ayudará a mantenerlo consistente de tal manera que todo el equipo pueda leerlo y reestructurarlo. El código que tiene el mismo aspecto fomenta la propiedad colectiva.

2.1.1.3.3 Codificar primero las Pruebas de Unidad

Crear las pruebas, antes del código, resulta mucho más fácil y más rápido para crear código. Beck (2000), afirma que el tiempo combinado que se necesita para crear una prueba unitaria y crear algo de código para hacerlo pasar es casi lo mismo de codificación. Pero, si ya se tiene las pruebas unitarias no se necesita crear después el código, lo que ahorrará bastante tiempo.

La creación de una unidad de prueba ayuda a un desarrollador en considerar realmente lo que hay que hacer. También ayuda a tener una respuesta inmediata mientras se trabaja. A menudo no está claro cuando un desarrollador ha terminado toda la funcionalidad necesaria. Si se crean las pruebas de unidad primero entonces se sabrá cuándo se ha terminado.

El código que va a crear es simple y conciso que implemente sólo las características que se quería. Otros desarrolladores pueden ver cómo utilizar este nuevo código, navegando por las pruebas.

2.1.1.3.4 Programación en parejas

Beck (2000) sostiene que todo el código que se enviará a la producción es creada por dos personas que trabajan juntas en un solo equipo. La programación en parejas incrementa la calidad del software y sin impactar en el tiempo de entrega. Es contrario a la intuición, pero 2 personas que trabajan en un solo equipo agregará tanta funcionalidad como dos trabajando por separado, excepto que será mucho más alto en calidad. Con el aumento de la calidad tiene un gran ahorro adelante en el proyecto.

La mejor manera de emparejar es sentarse al lado del otro en la parte frontal del monitor. Trabajar en pares de programación es una habilidad social que necesita tiempo para aprender. Los mejores programadores par saben cuándo decir "vamos a probar su idea en primer lugar."

Para Jeffries (2000), la programación en parejas no es tutoría. Una relación profesor-estudiante se siente muy diferente a diferencia de dos personas que trabajan juntas como iguales, incluso si uno tiene mucho más experiencia. Se necesita tiempo para acostumbrarse a la programación en parejas, así que no hay que preocuparse si existe incomodidad al principio.

2.1.1.3.5 Integración Secuencial

Se quiere que los desarrolladores sean capaces de avanzar en paralelo, realizar cambios en cualquier parte del sistema, pero también quiere una integración libre de error y que no se pierdan cambios. Al igual que una docena de locomotoras humeantes se dirigieran al patio de maniobras, al mismo tiempo. En lugar de restringir el desarrollo de naturaleza secuencial las locomotoras pueden entrar en el patio de maniobras sin un accidente

si simplemente se turnan. Se debe realizar esto con la integración de código también.

Estrictamente secuencial, la integración de los propios desarrolladores en combinación con la propiedad colectiva del código es una solución simple a este problema. Todo nuevo código se libera al repositorio de código fuente por turnos. Es decir, sólo un par de desarrollo integra, prueba y confirma los cambios en cualquier momento dado.

Esto no quiere decir que no se pueda integrar sus propios cambios con la última versión en su propia estación de trabajo. Simplemente no puede liberar a los cambios realizados en el equipo sin tener que esperar su turno.

Jeffries (2000) explica que el desarrollo avanza en paralelo, mientras que la integración es secuencial. Parece contra intuitivo para muchos desarrolladores ya que su experiencia es que la integración se tarda una eternidad por lo que se debe hacer pocas veces sea posible.

Se requiere algún tipo de mecanismo de secuenciación. Lo más sencillo es un solo equipo dedicado a la integración. Los pares de desarrollo secuencian implícitamente y se turnan para usarlo.

2.1.1.3.6 Integración a menudo

Los desarrolladores deben estar integrando e ingresar código en el repositorio de código cada hora, siempre que sea posible. En cualquier caso nunca aferrarse a cambios por más de un día. La integración continua con frecuencia, evita los esfuerzos de desarrollo divergentes o fragmentados, donde los desarrolladores no se comunican entre sí acerca de lo que se puede volver a utilizar, o lo que podría ser compartida. Todo el mundo tiene que trabajar con la última versión. No se deben hacer cambios en el código obsoleto causando dolores de cabeza de integración.

Jeffries (2000), asegura que cada par de desarrollo es responsable de la integración de su propio código. Esto puede ser cuando la unidad de prueba funciona al 100% o alguna parte más pequeña de la funcionalidad planeada está terminada.

La integración continua evita o detecta problemas de compatibilidad antes de tiempo. Es decir, se integra todo el proyecto en pequeñas cantidades.

2.1.1.3.7 Computadora dedicada a la Integración

Jeffries (2000) puntualiza que una computadora dedicada a versiones secuenciales actúa como una muestra física para controlar las entregas. El ordenador permite a los desarrolladores ver qué se está liberando y cuándo, así también garantiza la estabilidad cuando ya no hay cambios.

Debido a que una sola computadora utiliza el conjunto de pruebas está siempre al día. Si las pruebas de unidad funcionan al 100% se lograron los cambios sino los cambios se depuran o se respaldan en la estación de trabajo con los desarrolladores.

2.1.1.3.8 Propiedad Colectiva

La propiedad colectiva anima a todos a contribuir con nuevas ideas para todos los segmentos del proyecto. Según Beck (2000) cualquier desarrollador puede cambiar cualquier línea de código para agregar funcionalidades, corregir errores, mejorar los diseños o reestructurar. Una sola persona no se convierte en un cuello de botella para los cambios.

La forma en que esto funciona es que cada desarrollador cree pruebas unitarias para el código a medida que se desarrolla. Todo el código que se libera en el repositorio incluye pruebas unitarias que se ejecutan al cien por ciento.

Antes de la publicación de cualquier código debe pasar por todo el conjunto de pruebas. Así cualquier persona puede hacer un cambio a cualquier método de cualquier clase y liberarlo al repositorio de código, según sea necesario.

En la práctica, la propiedad colectiva es en realidad más fiable que poner una sola persona encargada de velar por las clases, sobretodo porque una persona puede dejar el proyecto en cualquier momento.

2.1.1.4 Pruebas

2.1.1.4.1 Pruebas Unitarias

Según Wells (1999), las pruebas unitarias son una de las piedras angulares de la Programación Extrema (XP). Los métodos “get” y “set” normalmente se omiten. Las pruebas unitarias se liberan en el repositorio de código junto con el código que ponen a prueba. El código sin pruebas no puede ser liberado. Si se descubre una prueba unitaria que falta debe ser creado en ese momento.

La mayor resistencia a dedicar esta cantidad de tiempo para las pruebas unitarias es una fecha límite que se acerca rápidamente. Lo más difícil de la pruebas es escribir lo que necesita para ahorros posteriores. Las pruebas unitarias automatizadas ofrecen un pago retroactivo mucho mayor que el costo de la creación.

Otro error común es que las pruebas unitarias se puedan escribir en los últimos tres meses del proyecto. Por desgracia, sin el desarrollo de pruebas unitarias se comen los últimos tres meses y luego algunos. Incluso las pruebas unitarias necesitan tiempo para evolucionar. El descubrimiento de todos los problemas que pueden ocurrir toma tiempo.

Wells (1999), afirma que las pruebas unitarias permiten la propiedad colectiva. Exigir que todo el código pase todas las pruebas unitarias antes de que pueda ser puesto en libertad asegura funcionalidad.

Las pruebas unitarias también permiten reestructuración. Después de cada pequeño cambio de las pruebas unitarias se pueden verificar que un cambio en la estructura no introdujo un cambio en la funcionalidad.

A menudo, la adición de nuevas funcionalidades requerirá el cambio de las pruebas unitarias para reflejar la funcionalidad.

Si bien es posible introducir un error en el código y probarlo rara vez ocurre en la práctica real. Ocasionalmente ocurren que la prueba está mal, pero el código es correcto. Esto se pone de manifiesto cuando el problema se investiga y se fija. La creación de pruebas independientes de código, con suerte antes del código, establece controles y equilibrios mejorando en gran medida las posibilidades de hacer las cosas bien la primera vez.

2.1.1.4.2 Pruebas de Aceptación

Las pruebas de aceptación se crean a partir de las historias de usuario. Las historias de usuario seleccionadas durante la reunión de planificación de la iteración se traducirán a las pruebas de aceptación. El usuario especifica escenarios para poner a prueba cuándo una historia de usuario se ha implementado correctamente. Wells (1999), menciona que una historia puede tener una o varias pruebas de aceptación, lo que sea necesario para garantizar las obras de funcionalidad.

Las pruebas de aceptación son las pruebas del sistema (caja negra). Cada prueba de aceptación representa algún

resultado esperado del sistema. Wells (1999), afirma que los usuarios son responsables de verificar la exactitud de las pruebas de aceptación y revisión de resultados de las pruebas para decidir qué no pasaron las pruebas y son de mayor prioridad. Las pruebas de aceptación también se utilizan como pruebas de regresión antes de una versión de producción.

Según Wells (1999), una historia de usuario no se considera completa hasta que haya pasado sus pruebas de aceptación. Esto significa que las nuevas pruebas de aceptación se deben crear cada iteración o el equipo de desarrollo reportará en cero los avances.

La calificación de la prueba de aceptación se publica en al equipo. Es responsabilidad del equipo programar el tiempo de cada iteración para arreglar las pruebas fallidas.

Las pruebas de aceptación cambiaron el nombre a partir de pruebas funcionales. Esto refleja mejor la intención, que es garantizar que los requisitos de los usuarios se cumplan y que el sistema sea aceptable.

“XP, delega una gran responsabilidad en las pruebas del software, por lo que se dice que sigue un desarrollo dirigido por pruebas, de tal manera que se asegura la calidad del producto según los requisitos capturados durante el desarrollo.” (Beck, 2000)

> Los creadores de XP aseguran el éxito de esta metodología en proyectos de alto riesgo, con una fecha de terminación fija, en los cuales el equipo de desarrollo no tenga una experiencia anterior similar e incluso en proyectos innovadores; plantean que XP ha sido creada para mitigar esos riesgos e incrementar la posibilidad de éxito. (Smith, 2001)

Finalmente XP se recomienda ampliamente en proyectos con equipos pequeños de desarrollo y requisitos muy cambiantes.

CAPÍTULO III

RESULTADOS

3.1 Planificación

La planificación se la ejecuta en reuniones entre el equipo de desarrollo y el usuario.

3.1.1 Historias de Usuario

Para el desarrollo de las historias de usuarios se realizaron reuniones entre la Dirección de Talento Humano y el equipo de desarrollo de tal manera que se indique la especificación de requerimientos.

3.1.1.1 Historias de Usuario (todos los usuarios)

Para la elaboración de las historias de usuarios de este proyecto se considerará los referentes expuestos en la Tabla 3.1.

Historia de Usuario	
Número: Identificador unívoco.	**Usuario:** Responsable de descripción.
Nombre de Historia: Tarea que se especifica.	
Prioridad en Negocio: Grado de prioridad para el desarrollo.	**Riesgo de Desarrollo:** En satisfacer los requerimientos del cliente.
Puntos estimados: Duración	**Iteración Asignada:** Orden de implementación.
Descripción: Detalle de lo que se realizará en la historia de usuario.	
Observaciones: Aclaraciones de un requerimiento que debería tomarse en cuenta.	

Tabla 3.1: Referentes de Historia de Usuario.

Elaborado: Altamirano, 2014

A continuación las historias de usuario que forman parte del presente trabajo:

Historia de Usuario			
Número:	1	**Usuario:**	Todos
Nombre de Historia:	Uso del servicio.		
Prioridad en Negocio:	ALTA	**Riesgo de Desarrollo:**	MEDIO
Puntos estimados:	0,5	**Iteración Asignada:**	1
Descripción: Todos los usuarios queremos usar el correo electrónico y una aplicación web, para ser notificados acerca del registro de asistencia.			
Observaciones: El correo electrónico es el institucional.			

Tabla 3.2: Historia de Usuario 1.

Elaborado: Altamirano, 2014

Historia de Usuario			
Número:	2	**Usuario:**	Todos
Nombre de Historia:	Acceso a la aplicación web.		
Prioridad en Negocio:	ALTA	**Riesgo de Desarrollo:**	MEDIO
Puntos estimados:	1,5	**Iteración Asignada:**	1
Descripción: Todos los usuarios queremos acceder a la aplicación web autenticando el correo electrónico institucional y la contraseña personal, para verificar la identidad.			
Observaciones: La contraseña por defecto es el número de cédula del usuario y pueden ser cambiada.			

Tabla 3.3: Historia de Usuario 2.

Elaborado: Altamirano, 2014

Historia de Usuario			
Número:	3	**Usuario:**	Todos
Nombre de Historia:	Consulta de inexistencia de registros.		
Prioridad en Negocio:	MEDIA	**Riesgo de Desarrollo:**	MEDIO
Puntos estimados:	1,5	**Iteración Asignada:**	3
Descripción: Todos los usuarios queremos consultar en el sistema web las inexistencia de registros, para una justificación a tiempo o aceptación por defecto.			
Observaciones: Las consultas son en un rango de fechas y exportables a un formato csv.			

Tabla 3.4: Historia de Usuario 3.

Elaborado: Altamirano, 2014

Historia de Usuario			
Número:	4	**Usuario:**	Todos
Nombre de Historia:	Consulta de registros impares.		
Prioridad en Negocio:	MEDIA	**Riesgo de Desarrollo:**	MEDIO
Puntos estimados:	1,5	**Iteración Asignada:**	3
Descripción: Todos los usuarios queremos consultar en el sistema web los registros impares notificados, para una justificación a tiempo o aceptación por defecto.			
Observaciones: Las consultas son en un rango de fechas y exportables a un formato csv.			

Tabla 3.5: Historia de Usuario 4.

Elaborado: Altamirano, 2014

Historia de Usuario			
Número:	5	**Usuario:**	Todos
Nombre de Historia:	Consulta de minutos no laborados.		
Prioridad en Negocio:	MEDIA	**Riesgo de Desarrollo:**	MEDIO
Puntos estimados:	1,5	**Iteración Asignada:**	3
Descripción: Todos los usuarios queremos consultar en el sistema web los minutos no laborados notificados, para una justificación a tiempo o aceptación por defecto.			
Observaciones: Las consultas son en un rango de fechas y exportables a un formato csv.			

Tabla 3.6: Historia de Usuario 5.

Elaborado: Altamirano, 2014

3.1.1.2 Historias de Usuario (Director de Talento Humano)

Historia de Usuario			
Número:	6	**Usuario:**	DTH
Nombre de Historia:	Notificación de inexistencia de registros.		
Prioridad en Negocio:	ALTA	**Riesgo de Desarrollo:**	ALTO
Puntos estimados:	2	**Iteración Asignada:**	2
Descripción: Como Director de Talento Humano (DTH) quiero almacenar y notificar automaticamente por correo electrónico a los trabajadores de la PUCESA sobre inexistencia de registros, para su justificación a tiempo o aceptación por defecto.			
Observaciones: Las notificaciones serán enviadas a la cuenta de correo electrónico institucional diariamente.			

Tabla 3.7: Historia de Usuario 6.

Elaborado: Altamirano, 2014

Historia de Usuario			
Número:	7	**Usuario:**	DTH
Nombre de Historia:	Notificación de registros impares.		
Prioridad en Negocio:	ALTA	**Riesgo de Desarrollo:**	ALTO
Puntos estimados:	2	**Iteración Asignada:**	2
Descripción: Como Director de Talento Humano (DTH) quiero almacenar y notificar automaticamente por correo electrónico a los trabajadores de la PUCESA sobre registros impares, para su justificación a tiempo o aceptación por defecto.			
Observaciones: Las notificaciones serán enviadas a la cuenta de correo electrónico institucional diariamente.			

Tabla 3.8: Historia de Usuario 7.

Elaborado: Altamirano, 2014

Historia de Usuario			
Número:	8	**Usuario:**	DTH
Nombre de Historia:	Notificación de minutos no laborados.		
Prioridad en Negocio:	ALTA	**Riesgo de Desarrollo:**	ALTO
Puntos estimados:	2	**Iteración Asignada:**	2
Descripción: Como Director de Talento Humano (DTH) quiero almacenar y notificar automaticamente por correo electrónico a los trabajadores de la PUCESA sobre minutos no laborados, para su justificación a tiempo o aceptación por defecto.			
Observaciones: Las notificaciones serán enviadas a la cuenta de correo electrónico institucional diariamente.			

Tabla 3.9: Historia de Usuario 8.

Elaborado: Altamirano, 2014

3.1.2 Estimación de Historias de Usuario

La valoración de las historias de usuario se realizaron por el equipo de desarrollo a través de reuniones de planificación.

Modulo	No.	Historias de Usuario	Prioridad	Riesgo	Tiempo Estimado		
					Semanas	Días	Horas
Aplicación	1	Uso del servicio.	ALTA	MEDIO	0,5	3	15
Mi cuenta	2	Acceso a la aplicación web.	ALTA	MEDIO	1,5	9	45
Reportes	3	Consulta de inexistencia de registros.	MEDIA	MEDIO	1,5	9	45
	4	Consulta de registros impares.	MEDIA	MEDIO	1,5	9	45
	5	Consulta de minutos no laborados.	MEDIA	MEDIO	1,5	9	45
Notificaciones	6	Notificación de inexistencia de registros.	ALTA	ALTO	2	12	60
	7	Notificación de registros impares.	ALTA	ALTO	2	12	60
	8	Notificación de minutos no laborados.	ALTA	ALTO	2	12	60

Tabla 3.10: Estimación de Historias de Usuario.

Elaborado: Altamirano, 2014

3.1.3 Plan de Entregas

Para elaborar el plan de entregas del presente proyecto las historias de usuario fueron agrupadas en entregables determinado tiempo calendario, esfuerzo de desarrollo e iteración.

3.1.3.1 Tiempo Calendario

El tiempo calendario (véase Tabla 3.11) especificó horas, días y semanas por mes lo que permitió calcular el esfuerzo de desarrollo posteriormente.

Tiempo Calendario		
Semanas por mes	Dias por semana	Horas por día
4	6	5

Tabla 3.11: Tiempo Calendario.

Elaborado: Altamirano, 2014

3.1.3.2 Esfuerzo de Desarrollo en Base a una Persona.

Se consideró a los integrantes del equipo de desarrollo, por lo tanto:

Personas en el Equipo	Horas de Esfuerzo de Desarrollo	Días de Esfuerzo de Desarrollo	Semana de Esfuerzo de Desarrollo
1 persona	5 horas	6 días	1 semana

Tabla 3.12: Esfuerzo de Desarrollo en Base a una Persona.

Elaborado: Altamirano, 2014

3.1.3.3 Plan de Entregas.

Modulo	No.	Historias de Usuario	Esfuerzo de Desarrollo			Calendario Estimado			Iteración Asiganda		
			Semanas	Días	Horas	Semanas	Días	Horas	1	2	3
Aplicación	1	Uso del servicio.	0,5	3	15	0,5	3	15	X		
Mi cuenta	2	Acceso a la aplicación web.	1,5	9	45	1,5	9	45	X		
Reportes	3	Consulta de inexistencia de registros.	1,5	9	45	1,5	9	45			X
	4	Consulta de registros impares.	1,5	9	45	1,5	9	45			X
	5	Consulta de minutos no laborados.	1,5	9	45	1,5	9	45			X
Notificaciones	6	Notificación de inexistencia de registros.	2	12	60	2	12	60		X	
	7	Notificación de registros impares.	2	12	60	2	12	60		X	
	8	Notificación de minutos no laborados.	2	12	60	2	12	60		X	

Tabla 3.13: Plan de Entregas.

Elaborado: Altamirano, 2014

3.2 Diseño

Para una visión más clara de la implementación de la aplicación, el equipo de desarrollo determinó las metáforas de la aplicación que describen las funcionabilidades usando un vocabulario técnico y entendible al usuario.

3.2.1 Metáfora de la Aplicación

La metáfora de la aplicación “NOTIFICACIÓN DEL REGISTRO DE ASISTENCIA PARA LOS TRABAJADORES DE LA PUCESA” está compuesto por la descripción técnica de los módulos que lo conforman: Modulo de “Aplicación”, Módulo de “Mi Cuenta”, Módulo de “Notificaciones” y Módulo de “Reportes”.

3.2.1.1 Módulo de “Aplicación”

Presenta la aplicación al usuario e informa sobre las operaciones que puede ejecutar.

3.2.1.2 Módulo de “Mi Cuenta”

Gestiona el cambio de contraseña asociada a la cuenta del usuario.

3.2.1.3 Módulo de “Notificaciones”

En modo administrador gestiona inexistencia de registros, registros impares y minutos no laborados que fueron notificados al correo electrónico institucional de todos los trabajadores de la PUCESA.

3.2.1.4 Módulo de “Reportes”

En modo administrador consulta entre dos fechas inexistencia de registros, registros impares y minutos no laborados que fueron notificados al correo electrónico institucional de todos los trabajadores de la PUCESA. Permite la exportación a formato CSV.

En modo trabajador consulta entre dos fechas inexistencia de registros, registros impares y minutos no laborados

que fueron notificados al correo electrónico institucional del propietario de la cuenta.

Permite la exportación a formato CSV.

3.2.2 Diagrama de Clases

La Ilustración 3.1 proporciona información sobre la estructura de la aplicación, mostrando: clases, atributos y relaciones.

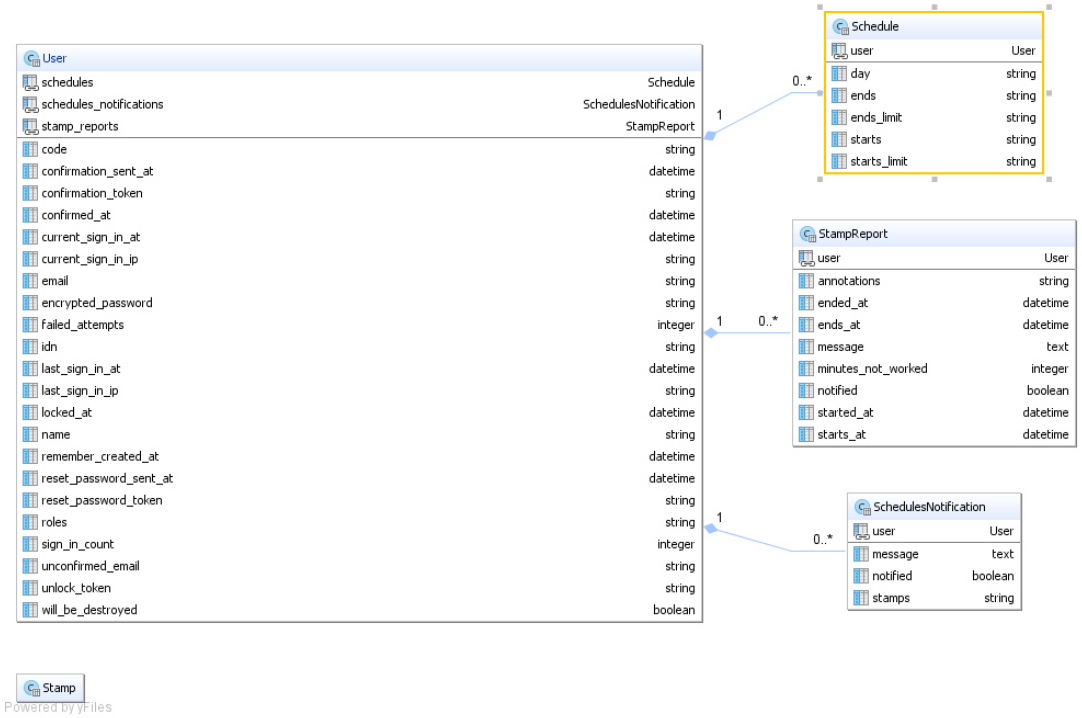


Ilustración 3.1: Diagrama de Clases.

Elaborado: Altamirano, 2014

3.2.3 Diseño de las Tarjetas CRC

Las tarjetas (Clase, Responsabilidad y Colaborador) facilitaron información al equipo de desarrollo acerca de cada una de las clases que se implementaron en la aplicación web.

User	
Responsabilidades	**Colaboradores**
Sincroniza horarios y usuarios. Gestiona usuarios y roles.	Shedule. StampReport. SchedulesNotification.

Tabla 3.14: Tarjeta CRC_User

Elaborado: Altamirano, 2014

Stamp	
Responsabilidades	**Colaboradores**
Obtiene registros del archivo de texto plano.	Ninguno.

Tabla 3.15: Tarjeta CRC_Stamp

Elaborado: Altamirano, 2014

Schedule	
Responsabilidades	**Colaboradores**
Genera reportes en horarios.	User.

Tabla 3.16: Tarjeta CRC_Schedule

Elaborado: Altamirano, 2014

StampReport	
Responsabilidades	**Colaboradores**
Analiza los registros y horarios. Agrega anotaciones en reportes.	User.

Tabla 3.17: Tarjeta CRC_StampReport

Elaborado: Altamirano, 2014

SchedulesNotification	
Responsabilidades	**Colaboradores**
Controla notificaciones al usuario.	User.

Tabla 3.18: Tarjeta CRC_SchedulesNotification

Elaborado: Altamirano, 2014

3.2.4 Diseño Arquitectónico

El diseño arquitectónico de la aplicación web se basa en la definición planteada por el "framework" Ruby on Rails (véase Ilustración 3.2).

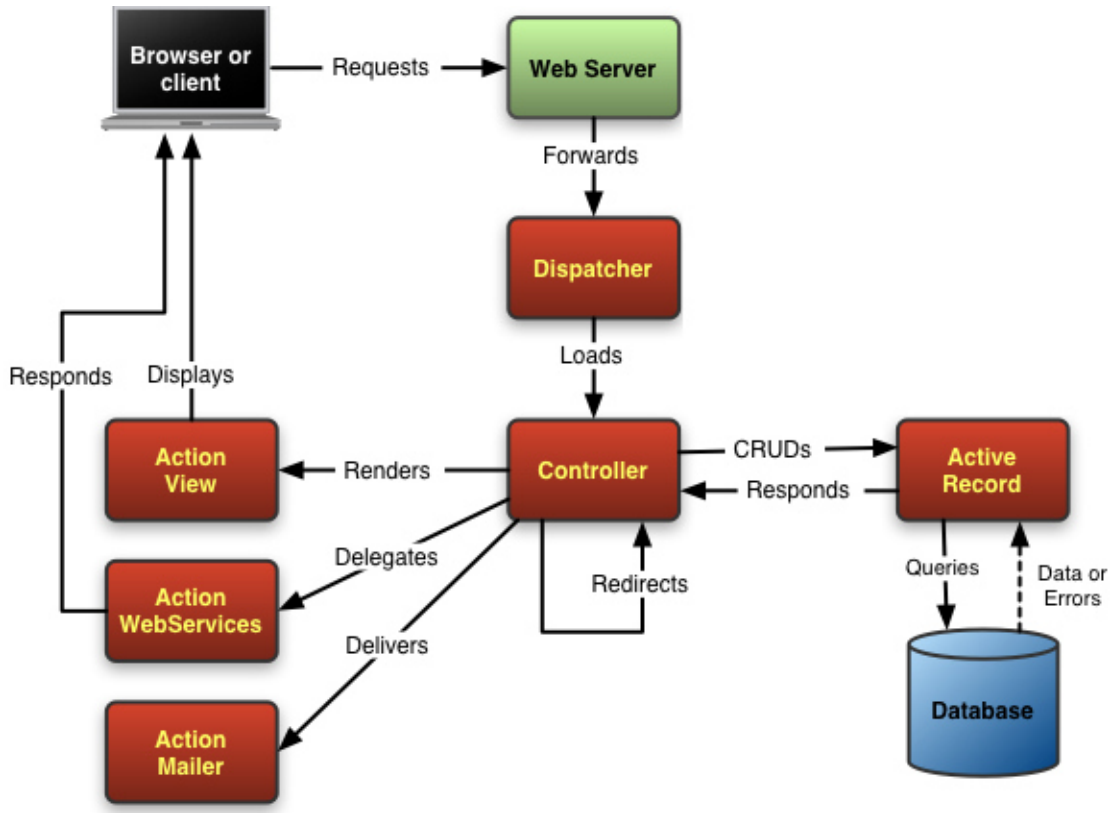


Ilustración 3.2: Arquitectura de la Aplicación Web

Fuente: http://www.rubyonrails.org.es

Como se puede observar la arquitectura de la aplicación en el existente proyecto funciona en el siguiente orden:

1. El explorador emite una solicitud.
2. Rails enruta la acción al controlador.
3. La acción pide el modelo para recuperar toda la información.
4. El modelo obtiene toda la información de la base de datos.
5. El modelo devuelve la información con el controlador.
6. El controlador captura la información, que se pasa a la vista ó envío de correos.
7. La vista representar la página como HTML.
8. “Action Mailer” envía correos.

3.2.5 Diseño de Interfaces

La Ilustración 3.3 evidencia la distribución de los elementos en la aplicación web.

Ilustración 3.3: Diseño de Interfaz

Elaborado: Altamirano, 2014

3.2.6 Estructura Jerárquica de la Aplicación Web

La Ilustración 3.4 muestra la organización jerárquica de las páginas web de la aplicación, organizadas en módulos y submódulos de acuerdo a la clasificación de las historias de usuario.

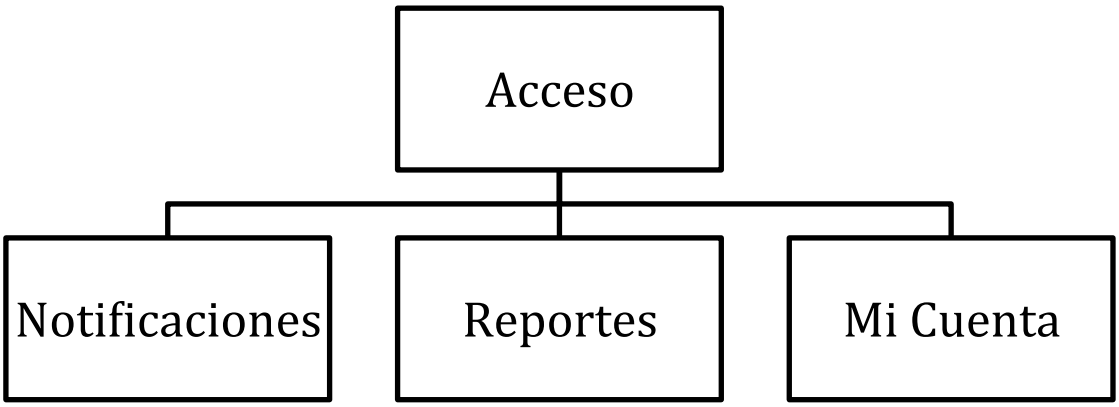


Ilustración 3.4: Estructura Jerárquica de la Aplicación Web.

Elaborado: Altamirano, 2014

3.3 Implementación

El equipo de desarrollo implementó las historias de usuario definidas.

3.3.1 Cronograma de Implementación de Iteraciones

Junto a las tareas que se ejecutaron se planificó la fecha de inicio y fin de implementación.

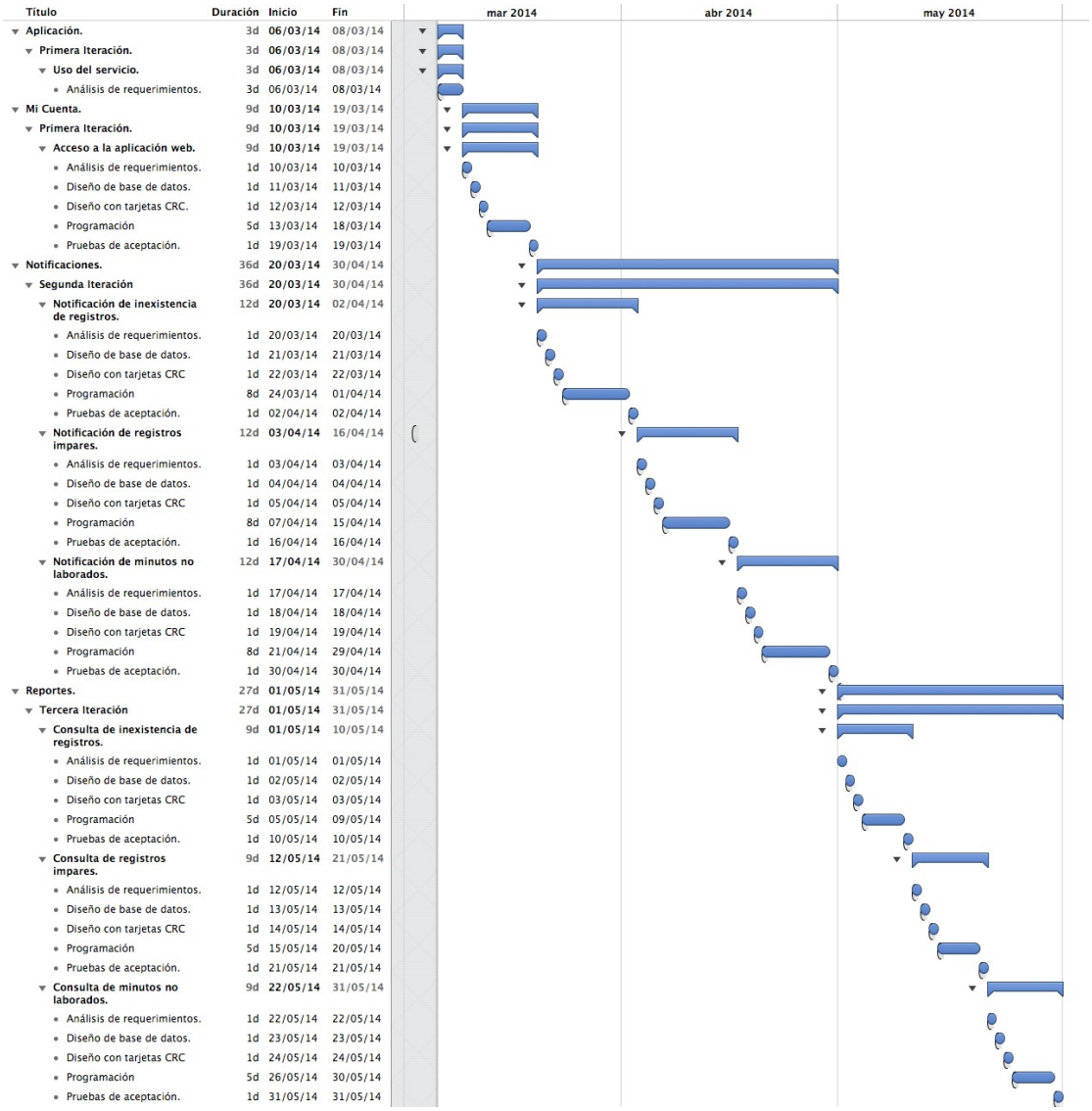


Ilustración 3.5: Cronograma de Iteraciones de Aplicaciones.

Elaborado: Altamirano, 2014

3.3.2 Implementación de Iteraciones.

El equipo de desarrollo en base al diseño definido implementó cada historia de usuario hasta cumplir las

pruebas de aceptación definidas, a continuación fragmentos de código:

users_controller.rb (controlador)

```
# Lista todos los usuarios del sistema.
def index
  @users = User.order('name ASC').page(params[:page])

  unless params[:name].blank?
    params[:name].split(/[ ]+/).tap do |words|
      @users.where!(
        "(#{ ( ['name LIKE ?'] * words.size ).join(' OR ') })",
        *words.collect{ |word| "%#{ word.upcase }%"}
      )
    end
  end

  @users.where!('code  LIKE ?',  "#{ params[:code].gsub(/\D+/, '') }%") unless params[:code].blank?

  @users.where!('idn   LIKE ?',  "#{ params[:idn].gsub(/\D+/, '') }%")  unless params[:idn].blank?

  @users.where!('email LIKE ?', "%#{ params[:email].downcase }%")      unless params[:email].blank?
```

```
  @users = @users.load
 end
```

stamp.rb (modelo)

```
  # Lee del archivo .log las marcas y las clasifica por codigo de usuario
  def all(stamps_path=nil)
    Hash.new do |hash, key|
      hash[key] = Hash.new{ |hash2, key2| hash2[key2] = [] }
    end.tap do |log|
      File.open(stamps_path || STAMPS_PATH) do |file|
        file.each_line do |line|
          add_date_to_log log, line.gsub(/[ ]+/, '').split(',') unless line.blank?
        end
      end
      log.each{ |_, stamps_by_user|  stamps_by_user.each{ |_, stamps| stamps.sort! ;
stamps.uniq! } }
    end
  end
  # Método para obtener la marca del archivo.log
  private
  def row_ary_to_date(ary)
    9.times{ ary.pop }
    y = 2000 + ary.pop.to_i
    d = ary.pop.to_i
    m = ary.pop.to_i
```

```
  mm = ary.pop.to_i
  hh = ary.pop.to_i
  new(y,m,d,hh,mm)
 end

 # Método para el codigo de usuario del archivo.log
 def row_ary_to_user_code(ary)
  ary.pop
  ary.pop.to_i.to_s
 end

 # Agrega una fecha a la base de logs
 def add_date_to_log(log, ary)
  date = row_ary_to_date(ary)
  log[date.to_date][row_ary_to_user_code(ary)] << date
 end
```

schedule.rb (modelo)

```
# Genera los reportes para este horario
def generate_reports!(stamps)
 user.stamp_reports.new do |report|
  ping_report(report, stamps)
        ping_report(report, stamps)
  report.analyze!
  report.save if report.has_incidents?
```

```
  end
 end

 def ping_report(report, stamps)
  stamps.dup.tap do |stamps_ary|
                stamps_ary.each do |stamp|
                 report.starts_limit = string_to_stamp(stamp, starts_limit)
                 report.starts_at = string_to_stamp(stamp, starts)
                 report.ends_at   = string_to_stamp(stamp, ends)
                 report.ends_limit   = string_to_stamp(stamp, ends_limit)
                 if report.ping!(stamp)
                        stamps.delete stamp
                        break
                 end
                end
  end
 end
```

stamp_report.rb (modelo)

```
# Calcula los tiempos y agrega anotaciones en el reporte
 def analyze!
  if message.blank?
   if started?
                  self.annotations << 'unclosed_session' unless closed?
          else
```

```
      self.annotations << 'absence'
      self.minutes_not_worked =  (ends_at.try :-, starts_at).to_i
    end
    self.minutes_not_worked /= 1.minute
    self.annotations_will_change!
  end
  self
 end

# Consulta si la sesión ha sido abierta.
def started?
  started_at.present?
 end
# Consulta si la sesión ha sido cerrada.
def closed?
  ended_at.present?
 end
# Abre una sesión.
 def start!(time=nil)
  return nil if started?
  self.started_at = time
  if started_at > starts_at
    self.annotations << 'delay'
    self.minutes_not_worked += started_at - starts_at
  end
  self
```

```
  end
# Cierra una sesion
  def close!(time=nil)
    return nil if closed? or !started?
    self.ended_at = time
    if ends_at > ended_at
      self.annotations << 'early_departure'
      self.minutes_not_worked += ends_at - ended_at
    end
    self
  end
# Intenta abrir o cerrar una sesión de acuerdo a la marca recibida.
  def ping!(stamp)
    ping(stamp, starts_limit.to_i, starts_at.to_i, ends_at.to_i, ends_limit.to_i)
  end

  def ping(stamp, starts_limit, starts_at, ends_at, ends_limit)
    case stamp.to_i
      when starts_limit..starts_at
        return start!(stamp)
      when starts_at..ends_at
        return self if close! stamp
        return start! stamp
      when ends_at..ends_limit
        close! stamp
      else
```

```
      nil
    end
  end

# Consulta si han habido incidentes
  def has_incidents?
    annotations.any?
  end

  # Humaniza las anotaciones
  def details
    return message if message.present?
    [].tap do |str|
      str << I18n.t('stamp_reports.annotations.absence')          if
annotations.include?('absence')
      str << I18n.t('stamp_reports.annotations.delay')         if
annotations.include?('delay')
      str << I18n.t('stamp_reports.annotations.unclosed_session') if
annotations.include?('unclosed_session')
      str << I18n.t('stamp_reports.annotations.early_departure')      if
annotations.include?('early_departure')
    end.join('<br />').html_safe
  end
```

notifications_mailer (Mailer)

```
class NotificationsMailer < ActionMailer::Base
  default from: 'asistencia@pucesa.edu.ec'
```

```
  def reports(user, reports)
    @user = user
    @schedules = @user.schedules.to_a
    @reports = reports

    mail subject: "Minutos No Laborados #{@reports.first.try{ |report|
(report.starts_at || report.created_at).to_date.strftime('%d/%m/%Y') }}", to:
@user.email
  end

  def schedules(user, schedules, message="", stamps=[])
    @user      = user
    @schedules = schedules
    @message   = message
    @stamps    = stamps

    mail subject: @message, to: @user.email
  end
end
```

3.3.3 Seguimiento de las Iteraciones.

En el actual proyecto de desarrollo ágil (XP) un diagrama de quemado (Burn Down) representó gráficamente el progreso del equipo de desarrollo en intervalos de tiempo

con el objetivo de supervisar el cumplimiento del proyecto de acuerdo a la planificación así también permitió observar el esfuerzo real en comparación al esfuerzo estimado del equipo de desarrollo.

3.3.3.1 Seguimiento de la Primera Iteración.

BurnDown: Primera Iteración		
Fechas	Esfuerzo Estimado	Esfuerzo Real
06/03/14	5	4
10/03/14	5	4
11/03/14	5	4
12/03/14	5	4
13/03/14	25	20
19/03/14	5	4

Tabla 3.19: Seguimiento de la Primera Iteración.

Elaborado: Altamirano, 2014

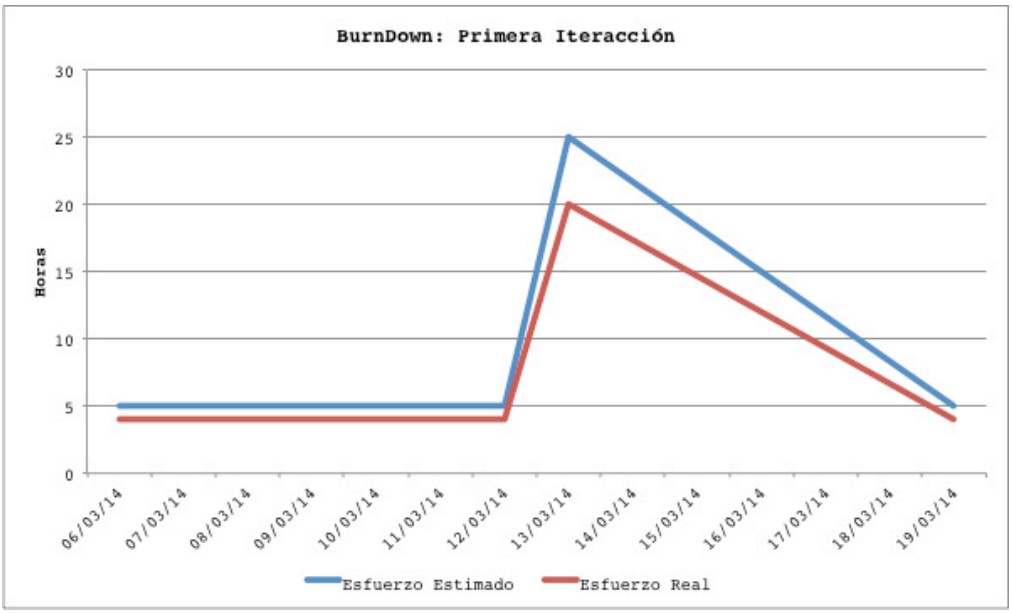


Ilustración 3.6: Burn Down de Primera Iteración.

Elaborado: Altamirano, 2014

3.3.3.2 Seguimiento de la Segunda Iteración.

BurnDown: Segunda Iteración		
Fechas	Esfuerzo Estimado	Esfuerzo Real
20/03/14	5	4
21/03/14	5	4
22/03/14	5	4
24/03/14	40	32
02/04/14	5	4
03/04/14	5	4
04/04/14	5	4
05/04/14	5	4
07/04/14	40	32
16/04/14	5	4
17/04/14	5	4
18/04/14	5	4
19/04/14	5	4
21/04/14	40	32
30/04/14	5	4

Tabla 3.20: Seguimiento de la Segunda Iteración.

Elaborado: Altamirano, 2014

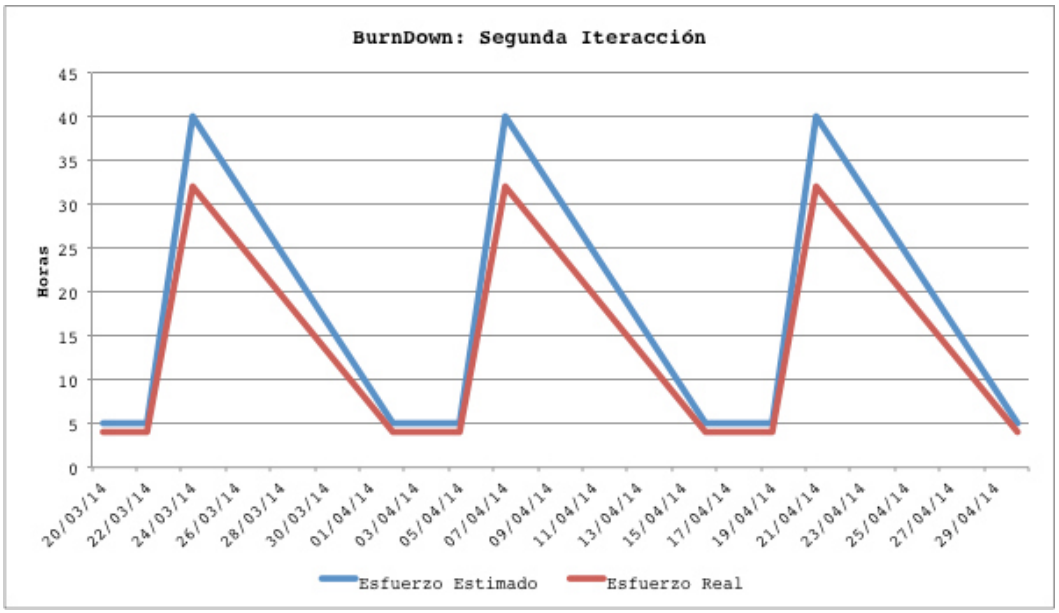


Ilustración 3.7: Burn Down de Segunda Iteración.

Elaborado: Altamirano, 2014

3.3.3.3 Seguimiento de la Tercera Iteración.

BurnDown: Tercera Iteración		
Fechas	Esfuerzo Estimado	Esfuerzo Real
02/05/14	5	4
03/05/14	5	4
05/05/14	25	20
10/05/14	5	4
12/05/14	5	4
13/05/14	5	4
14/05/14	5	4
15/05/14	25	20
21/05/14	5	4
22/05/14	5	4
23/05/14	5	4
24/05/14	5	4
26/05/14	25	20
31/05/14	5	4

Tabla 3.21: Seguimiento de la Tercera Iteración.

Elaborado: Altamirano, 2014

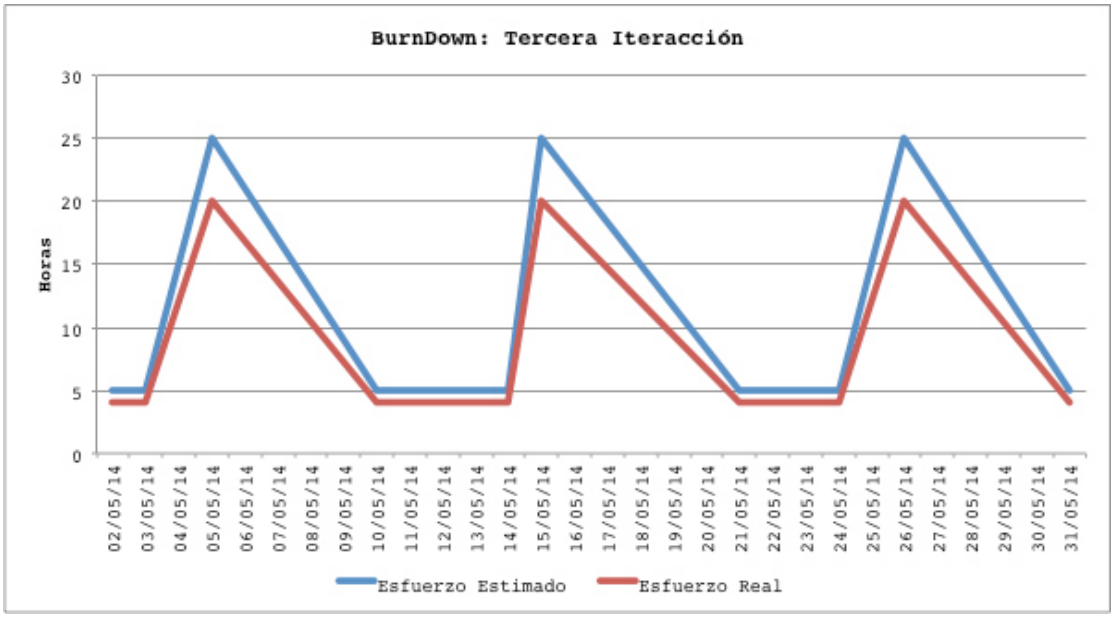


Ilustración 3.8: Burn Down de Tercera Iteración.

Elaborado: Altamirano, 2014

Como se observa en las ilustraciones 3.6, 3.7 y 3.8 el esfuerzo real es menor al estimado durante toda las iteraciones debido a que el equipo de desarrollo adquirió experiencia en el uso de la tecnología de desarrollo, mejoró la comunicación con el cliente y se reutilizó gran parte del código que se desarrolló en las anteriores iteraciones.

3.4 Pruebas

3.4.1 Pruebas Unitarias

La retroalimentación entre el diseño, corrección y la refactorización de código fue gracias a la ejecución de pruebas unitarias previo al desarrollo.

En esta ocasión se usó RSPEC para la ejecución de mencionadas pruebas. RSPEC es una herramienta que dirige y valida el desarrollo de una aplicación.

Adicionalmente se consideró las mejores prácticas que otros desarrolladores han aprendido a través de años de

experiencia expuestos en http://betterspecs.org/ (directrices RSPEC con Ruby), iniciado por Andrea Reginato.

user_spec.rb

```
require 'spec_helper'

class AnotherStruct < OpenStruct
  def as_json(format=nil)
    super()['table']
  end
end



describe User do
  let!(:user){ User.new }
  describe '#role_on!' do
    it { expect(user.role_on!(:user).role?(:user)).to
be_truthy }

    it { expect(user.role_on!(:user).role?(:user2)).to
be_falsey }
  end

  describe '#role_off!' do
```

```
    it {
expect(user.role_on!(:user).role_off!(:user).role?(:user)
).to be_falsey }


    it {
expect(user.role_on!(:user).role_off!(:user2).role?(:user
)).to be_truthy}
  end


  describe '#admin?' do
    it { expect(user.admin?).to be_falsey }


    it { expect(user.role_on!(:admin).admin?).to
be_truthy }


    it { expect(user.role_on!(:su).admin?).to be_truthy }


    it { expect(user.role_on!(:sua).admin?).to be_falsey}
  end


  describe '#su?' do
    it { expect(user.role_on!(:admin).su?).to be_falsey }


    it { expect(user.role_on!(:su).su?).to be_truthy }


    it { expect(user.role_on!(:sua).su?).to be_falsey }
  end


  context 'with employees integration' do
```

```
    let!(:employee){
      AnotherStruct.new(
          email: 'user@example.com',
          name:  'bbbbbbbb',
          code:  '0000000000',
          id:    '0000000000',
          schedules: [
              AnotherStruct.new(day: 'monday',  starts:
'0900', ends: '1300'),
              AnotherStruct.new(day: 'monday',  starts:
'1400', ends: '1600'),
              AnotherStruct.new(day: 'tuesday', starts:
'0900', ends: '1300'),
              AnotherStruct.new(day: 'tuesday', starts:
'1400', ends: '1600'),
          ]
      )
    }


    describe '#fix_schedules!' do
      it do
        user = User.from_employee(employee)
        user.schedules.create(day: 'tuesday', starts:
'1600', ends: '2200')
        user.fix_schedules!
        expect(user.schedules.where(day: 'tuesday',
starts: '1400', ends: '2200').exists?).to be_truthy
      end
    end
```

```
    describe '::from_employee' do
      it do
        employee.email = '100000'
        expect(User.from_employee(employee)).to be_nil
      end

      it do
        employee.id = nil
        expect(
            User.from_employee(employee).persisted?
        ).to be_falsey
      end

      it do
        expect(
            User.from_employee(
                employee
            ).persisted?
        ).to be_truthy
      end

      it do
        another_schedules = [
            AnotherStruct.new(day: 'monday',  starts:
'0900', ends: '1300'),
            AnotherStruct.new(day: 'monday',  starts:
'1400', ends: '1800'),
```

```
            AnotherStruct.new(day: 'tuesday', starts:
'0900', ends: '1300'),

            AnotherStruct.new(day: 'tuesday', starts:
'1400', ends: '1900'),

        ]

        user = User.from_employee(employee)

        user.sync_schedules!(another_schedules)

        user.schedules.reload

        expect(user.schedules.collect(&:id).sort).to
eq(another_schedules.collect{|schedule|
user.schedules.find_by(schedule.as_json).try :id}.sort)

      end

    end

  end

end
```

stamps_spec.rb

```
require 'spec_helper'


describe Stamp do

  describe '::DAYNAMES' do

    it{ expect(Stamp::DAYNAMES).to eq(%w(sunday monday
tuesday wednesday thursday friday saturday)) }

  end


  describe '::all' do

    it{ expect(Stamp.all('spec/data.log')[Date.new(2014,
7, 14)]['3373']).to eq( [ Stamp.new(2014, 7, 14, 7, 53),
```

```
Stamp.new(2014, 7, 14, 10, 2), Stamp.new(2014, 7, 14, 12,
0) ] ) }
  end


  describe '#day_name' do
    it { expect(Stamp.new(2014, 7, 14).day_name).to
eq('monday')    }


    it { expect(Stamp.new(2014, 7, 15).day_name).to
eq('tuesday')   }


    it { expect(Stamp.new(1936,12, 16).day_name).to
eq('wednesday') }


    it { expect(Stamp.new(1542, 2, 12).day_name).to
eq('thursday')  }


    it { expect(Stamp.new(1830,12, 17).day_name).to
eq('friday')    }


    it { expect(Stamp.new(1989, 7, 29).day_name).to
eq('saturday')  }


    it { expect(Stamp.new(1912, 1, 28).day_name).to
eq('sunday')    }
  end


  it('Hereda de la clase Time') { expect(Stamp.now).to
be_a(Time) }
end
```

stamp_report_spec.rb

```
require 'spec_helper'

describe StampReport do
  let!(:report){
    StampReport.new(
        starts_limit: Stamp.new(2014, 1, 1,  6, 0),
        starts_at:    Stamp.new(2014, 1, 1,  8, 0),
        ends_at:      Stamp.new(2014, 1, 1, 16, 0),
        ends_limit:   Stamp.new(2014, 1, 1, 23, 59)
    )
  }

  describe '#started?' do
    it{ expect(report.started?).to be_falsey }

    it do
      report.started_at = report.starts_at + 5.minutes
      expect(report.started?).to be_truthy
    end
  end

  describe '#closed?' do
    it{ expect(report.closed?).to be_falsey }

    it do
      report.ended_at = report.ends_at + 5.minutes
```

```
    expect(report.closed?).to be_truthy
  end
end

describe '#start!' do
  context 'Llega temprano' do
    let!(:started_report) do
      report.start!(report.starts_at - 1.hour)
      report
    end

    it { expect(report.started?).to be_truthy }

    it { expect(report.annotations).to eq([]) }

    it { expect(report.minutes_not_worked).to be(0) }
  end

  context 'Llega tarde' do
    let!(:started_report) do
      report.start!(report.starts_at + 10.minutes)
      report
    end

    it { expect(report.started?).to be_truthy }

    it { expect(report.annotations).to eq(['delay']) }
```

```
      it { expect(report.minutes_not_worked).to
be(10.minutes.to_i) }
    end
  end

  describe '#close!' do
    let!(:started_report) do
      report.start!(report.starts_at - 1.hour)
      report
    end

    context 'Sale temprano' do
      let!(:closed_report){ report.close!(report.ends_at
- 10.minutes) }

      it { expect(report.closed?).to be_truthy }

      it { expect(report.annotations).to
eq(['early_departure']) }

      it { expect(report.minutes_not_worked).to
be(10.minutes.to_i) }
    end

    context 'Sale tarde' do
      let!(:closed_report){ report.close!(report.ends_at
+ 10.minutes) }
```

```
      it { expect(report.closed?).to be_truthy }

      it { expect(report.annotations).to eq([]) }

      it { expect(report.minutes_not_worked).to be(0) }
    end
  end


  describe '#ping!' do
    context 'intenta marcar la entrada' do
      context 'marca fuera del rango' do
        it do
          report.ping!(report.starts_limit - 10.minutes)
          expect(report.started?).to be_falsey
        end

        it do
          report.ping!(report.ends_at + 10.minutes)
          expect(report.started?).to be_falsey
        end

        context 'intenta marcar la salida' do
          before { report.ping!(report.ends_at +
10.minutes) }

          it do
            report.ping!(report.ends_at + 10.minutes)
```

```
          expect(report.closed?).to be_falsey
        end
      end

    end

    context 'marca dentro del rango valido' do
      it do
        report.ping!(report.ends_at - 10.minutes)
        expect(report.started?).to be_truthy
      end

      it do
        report.ping!(report.starts_at - 10.minutes)
        expect(report.started?).to be_truthy
      end

      it do
        report.ping!(report.starts_at + 10.minutes)
        expect(report.started?).to be_truthy
      end


      context 'intenta marcar la salida' do
        before { report.ping!(report.starts_at -
10.minutes) }

        context 'dentro del rango' do
```

```
            it do
              report.ping!(report.ends_limit -
10.minutes)
              expect(report.closed?).to be_truthy
            end

            it do
              report.ping!(report.ends_limit -
10.minutes)
              expect(report.closed?).to be_truthy
            end

            it do
              report.ping!(report.ends_at + 10.minutes)
              expect(report.closed?).to be_truthy
            end

            it do
              report.ping!(report.ends_at - 10.minutes)
              expect(report.closed?).to be_truthy
            end
          end

          context 'fuera del rango' do
            it do
              report.ping!(report.ends_limit +
10.minutes)
              expect(report.closed?).to be_falsey
```

```
            end
          end
        end
      end
    end
  end
end
```

3.4.2 Pruebas de Aceptación

Con el fin de desarrollar únicamente lo necesario se definieron escenarios planteados entre el usuario y el equipo de desarrollo previo al desarrollo de cada historia de usuario.

A continuación se presentan las pruebas de aceptación ejecutadas para el presente proyecto.

Número de prueba:	1
Historia de usuario:	Uso del servicio.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
Entrada:	Ninguna.
Resultado Esperado 1:	Web Service actualizado.
Resultado Esperado 2:	Ninguno.

Tabla 3.22: Prueba de Aceptación 1.

Elaborado: Altamirano, 2014

Número de prueba:	2
Historia de usuario:	Acceso a la aplicación web.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario ingresa a la página de acceso.
	El usuario ingresa su correo electrónico institucional y la contraseña de la aplicación web.
Resultado Esperado 1:	La aplicación web permite el acceso a sus módulos.
Resultado Esperado 2:	La aplicación web muestra un mensaje de invalidéz.

Tabla 3.23: Prueba de Aceptación 2.

Elaborado: Altamirano, 2014

Número de prueba:	3
Historia de usuario:	Acceso a la aplicación web.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario ingresa a la página de acceso.
	El usuario ingresa su correo electrónico institucional.
	El usuario ingresa la contraseña de la aplicación web incorrectamente reiteradas veces.
Resultado Esperado 1:	La aplicación web, fallido el cuarto intento muestra un mensaje de advertencia acerca de un quinto y último intento caso contrario la cuenta se bloqueará.
Resultado Esperado 2:	La aplicación web bloquea la cuenta y envía una autorización de desbloqueo al correo electrónico institucional del usuario.

Tabla 3.24: Prueba de Aceptación 3.

Elaborado: Altamirano, 2014

Número de prueba:	4
Historia de usuario:	Acceso a la aplicación web.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario ingresa a la página de acceso.
	El usuario ingresa su correo electrónico institucional.
	El usuario ingresa la contraseña de la aplicación web incorrectamente reiteradas veces hasta bloquear la cuenta.
	El usuario selecciona el enlace "Desbloquear mi cuenta"
	El usuario ingresa su correo electrónico institucional.
	El usuario selecciona "Enviar instrucciones".
Resultado Esperado 1:	La aplicación web envía una autorización de desbloqueo al correo electrónico institucional del usuario
Resultado Esperado 2:	La aplicación web muestra un mensaje acerca que el usuario no está bloqueado.

Tabla 3.25: Prueba de Aceptación 4.

Elaborado: Altamirano, 2014

Número de prueba:	5
Historia de usuario:	Acceso a la aplicación web.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario ingresa a la página de acceso.
	El usuario ingresa su correo electrónico institucional.
	El usuario ingresa la contraseña de la aplicación web incorrectamente reiteradas veces hasta bloquear la cuenta.
	El usuario selecciona el enlace "Desbloquear mi cuenta"
	El usuario ingresa su correo electrónico institucional.
	El usuario selecciona "Enviar instrucciones".
	El usuario ingresa a su cuenta de correo electrónico.
	Selecciona la autorización que dice "Desbloquear mi cuenta"
Resultado Esperado 1:	La aplicación web desbloquea la cuenta y permite nuevos intentos de acceso.
Resultado Esperado 2:	La aplicación web muestra un mensaje acerca que la autorización no es válida.

Tabla 3.26: Prueba de Aceptación 5.

Elaborado: Altamirano, 2014

Número de prueba:	6
Historia de usuario:	Acceso a la aplicación web.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario ingresa a la página de acceso.
	El usuario ingresa su correo electrónico institucional.
	El usuario debido a un olvido involuntario de la contraseña selecciona el enlace "¿Olvidate tu contraseña?"
	El usuario ingresa su correo electrónico institucional.
	El usuario selecciona "Enviar instrucciones".
Resultado Esperado 1:	La aplicación web envía una autorización para cambiar la contraseña de la cuenta de la aplicación web.
Resultado Esperado 2:	La aplicación web muestra un mensaje acerca que el usuario (correo electrónico) no se encuentra.

Tabla 3.27: Prueba de Aceptación 6.

Elaborado: Altamirano, 2014

Número de prueba:	7
Historia de usuario:	Acceso a la aplicación web.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario ingresa a la página de acceso.
	El usuario ingresa su correo electrónico institucional.
	El usuario debido a un olvido involuntario de la contraseña selecciona el enlace "¿Olvidate tu contraseña?"
	El usuario selecciona el enlace "¿Olvidate tu contraseña?"
	El usuario ingresa su correo electrónico institucional.
	El usuario selecciona "Enviar instrucciones".
	El usuario ingresa a su cuenta de correo electrónico institucional.
	Selecciona la autorización que dice "Cambiar mi clave"
Resultado Esperado 1:	La aplicación web permite asignar una nueva contraseña de acceso.
Resultado Esperado 2:	Ninguno.

Tabla 3.28: Prueba de Aceptación 7.

Elaborado: Altamirano, 2014

Número de prueba:	8
Historia de usuario:	Acceso a la aplicación web.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario ingresa a la página de acceso.
	El usuario ingresa su correo electrónico institucional.
	El usuario selecciona el módulo "Mi Cuenta" en la aplicación web.
	El usuario ingresa la nueva contraseña con mínimo ocho caracteres.
	El usuario confirma la nueva contraseña.
	El usuario ingresa la actual contraseña.
Resultado Esperado 1:	La aplicación web cambia la contraseña de acceso a la cuenta.
Resultado Esperado 2:	La aplicación web muestra un mensaje acerca de la inadecuada longitud de la contraseña.
Resultado Esperado 3:	La aplicación web muestra un mensaje acerca de la inadecuada confirmación de la contraseña.
Resultado Esperado 4:	La aplicación web muestra un mensaje acerca de la inadecuada contraseña actual.

Tabla 3.29: Prueba de Aceptación 8.

Elaborado: Altamirano, 2014

Número de prueba:	9
Historia de usuario:	Notificación de inexistencia de registros.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario no ejecuta registros de asistencia en el presente día.
Resultado Esperado 1:	La aplicación web envía una notificación al correo electrónico institucional del usuario acerca de inexistencia de registros del presente día.
Resultado Esperado 2:	La aplicación web no almacena registros.

Tabla 3.30: Prueba de Aceptación 9.

Elaborado: Altamirano, 2014

Número de prueba:	10
Historia de usuario:	Notificación de registros impares.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario ejecuta registros impares de asistencia en el presente día.
Resultado Esperado 1:	La aplicación web envía una notificación al correo electrónico institucional del usuario acerca de registros impares del presente día.
Resultado Esperado 2:	La aplicación web muestra los registros impares.

Tabla 3.31: Prueba de Aceptación 10.

Elaborado: Altamirano, 2014

Número de prueba:	11
Historia de usuario:	Notificación de minutos no laborados.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario ejecuta registros con retraso en el presente día.
Resultado Esperado 1:	La aplicación web envía una notificación al correo electrónico institucional del usuario acerca de minutos no laborados por retraso del presente día.
Resultado Esperado 2:	La aplicación web muestra los minutos no laborados por retraso.

Tabla 3.32: Prueba de Aceptación 11.

Elaborado: Altamirano, 2014

Número de prueba:	12
Historia de usuario:	Notificación de minutos no laborados.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
Entrada:	El usuario ejecuta registros con salida temprana en el presente día.
Resultado Esperado 1:	La aplicación web envía una notificación al correo electrónico institucional del usuario acerca de minutos no laborados por salida temprana del presente día.
Resultado Esperado 2:	La aplicación web muestra los minutos no laborados por salida temprana.

Tabla 3.33: Prueba de Aceptación 12.

Elaborado: Altamirano, 2014

Número de prueba:	13
Historia de usuario:	Consulta de inexistencia de registros.
	Consulta de registros impares.
	Consulta de minutos no laborados.
Precondiciones:	Correo electrónico institucional asignado al usuario.
	Correo electrónico institucional de usuario registrado para notificaciones de asistencia.
	Usuario activo en la aplicación web.
	Usuario notificado por inexistencia de registros ó registros impares ó minutos no laborados.
Entrada:	El usuario selecciona el módulo de reportes.
	El usuario ingresa una fecha en "desde" y otra fecha en "hasta".
	El usuario selecciona "Buscar".
Resultado Esperado 1:	La aplicación web muestra las notificaciones acerca de registros inexistentes, registros impares y minutos no laborados comprendidos entre fechas.
Resultado Esperado 2:	Ninguno.

Tabla 3.34: Prueba de Aceptación 13.

Elaborado: Altamirano, 2014

3.5 Planificación del Despliegue

Debido a que el presente proyecto tiene como objetivo implementarlo (ponerlo en producción) se ha visto conveniente documentar este proceso como una fase más del proyecto mas no de la metodología que terminó con la documentación de pruebas.

Una vez terminado el desarrollo de la aplicación web, el equipo de desarrollo se reunió con el usuario para planificar el despliegue.

A continuación se muestra el despliegue que describe toda la información necesaria para cumplir con lo mencionado.

3.5.1 Requisitos Mínimos de Hardware para la Instalación.

3.5.1.1 Lector Biométrico.

- HandPunch 2000.

3.5.1.2 Servidor Web.

- Memoria Ram: 1Gb
- Disco Duro: 1 Gb (libre).
- Procesador: 1 Ghz.

3.5.1.3 Cliente.

- Memoria Ram: 512 Mb.
- Procesador: 1 Ghz

3.5.2 Requisitos Mínimos de Software para la Instalación

3.5.2.1 Servidor Web.

Sistema Operativo:

- Windows XP o Linux.

Aplicaciones Base:

- HP32.
- Tibbo Device Server Toolkit.

Sistema de Gestión de Base de Datos:

- PostgreSQL

Interprete:

- JRuby

Servidor de Aplicaciones:

- JBoss/Torquebox

3.5.2.2 Cliente.

Navegador Web:

- Internet Explorer, Mozilla Firefox, Google Chrome o equivalente sobre cualquier dispositivo gracias a la tecnología usada RWD (Responsive Web Design).

3.5.3 Despliegue.

3.5.3.1 Instalación y Configuración de la Base de Datos.

1. Instalar PostgreSQL con puerto 5432 y una contraseña.

2. Crear un usuario con privilegios de superadministrador y su respectiva contraseña.

3.5.3.2 Instalación del Intérprete de la Aplicación.

1. Instalar JRuby.

3.5.3.3 Copia de Archivos y Directorios.

1. Copiar el directorio “Aplicación” al disco duro directamente. Ej. D:\

2. Copiar el archivo “poll.exe” en C:\Program Files (x86)\Hand Punch32

3.5.3.4 Ejecución del Despliegue en Consola.

```
d:

cd d:\aplicacion\puce

gem install bundler

bundle install

gem install torquebox torquebox-server

rake db:create db:migrate db:seed RAILS_ENV=production
```

rake assets:precompile

torquebox deploy

Ejecutar el starserver.bat

3.5.3.5 Configuración de Tareas Programadas.

1. Configurar en el inicio del sistema: “start-server”

2. Configurar a partir de las 00h00 “sync.exe”

3. Configurar luego de las 10h00 y antes de las 00h00 “poll.exe”

CAPÍTULO IV
DISCUSIÓN / ANÁLISIS Y VALIDACIÓN DE LOS RESULTADOS

4.1 Análisis de Resultados

4.1.1 Resultado de la Implementación de la Primera Iteración

4.1.1.1 Resultado del Módulo "Aplicación"

La Ilustración 4.1 muestra el módulo "Aplicación" que como objetivo tiene informar sobre funcionalidades del servicio.

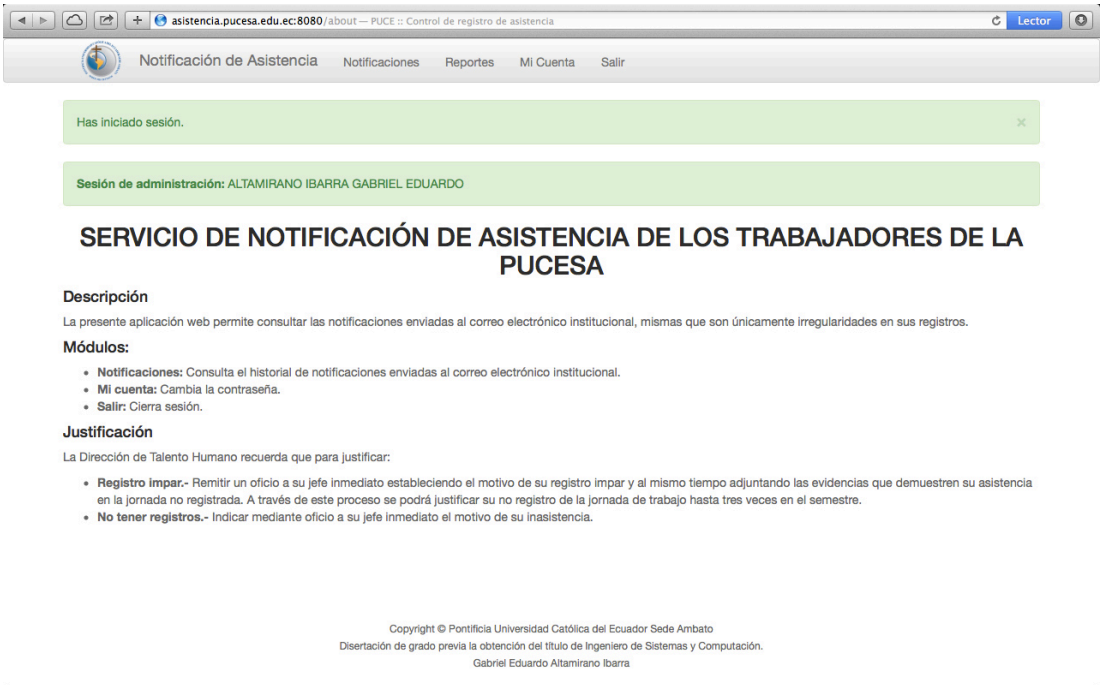


Ilustración 4.1: Módulo "Aplicación".

4.1.1.2 Resultado del Módulo "Mi Cuenta"

El módulo "Mi Cuenta" posee varias opciones que son útiles para la eficiente gestión de acceso a la aplicación web (véase Ilustración 4.2). La Ilustración 4.3 muestra el resultado de un acceso inválido ya sea por inconvenientes con el correo electrónico o contraseña.

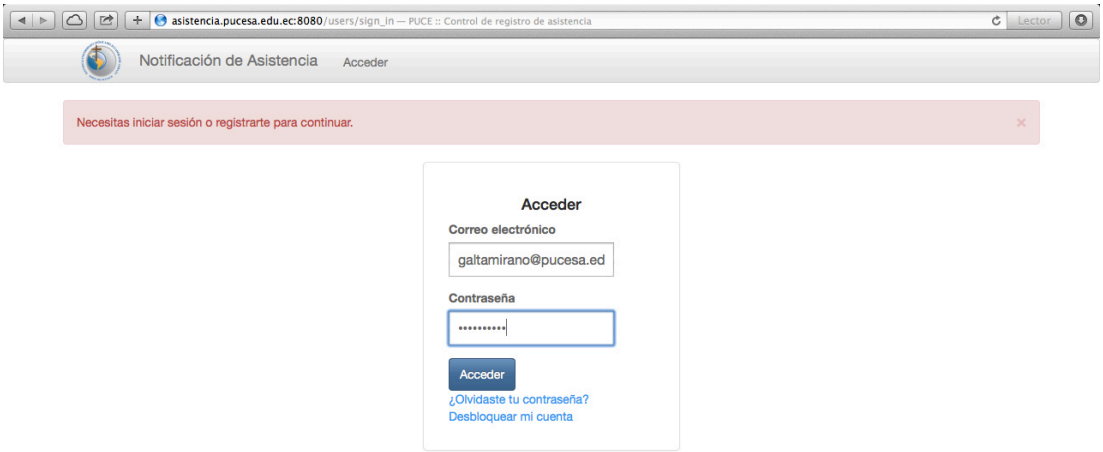


Ilustración 4.2: Acceso a la Aplicación Web.

Ilustración 4.3: Acceso Inválido a la Aplicación Web.

La Ilustración 4.4 muestra una alerta previa al bloqueo de la cuenta debido a dos intentos seguidos y fallidos de ingreso de la contraseña.

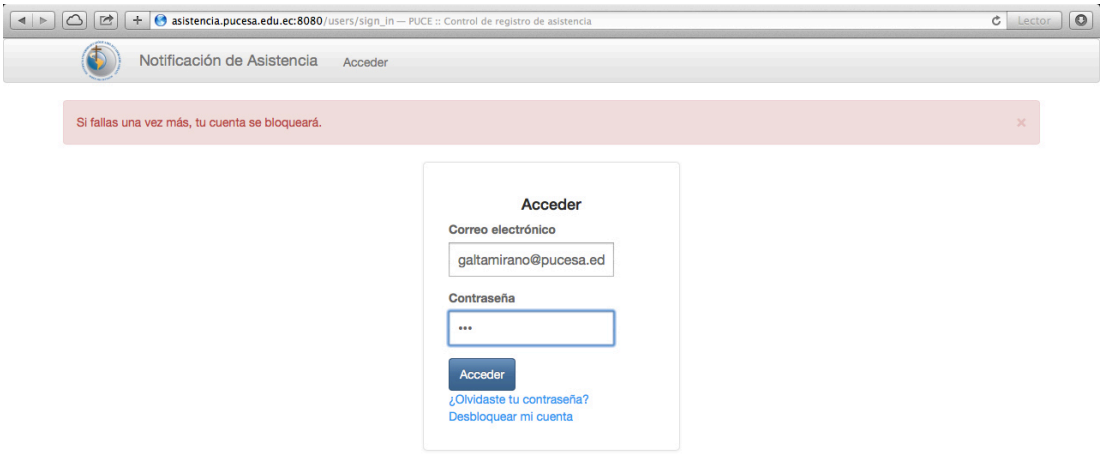


Ilustración 4.4: Alerta de Boqueo de Cuenta.

La Ilustración 4.5 muestra el bloqueo de la cuenta debido a un tercer intento seguido y fallido de ingreso de contraseña.

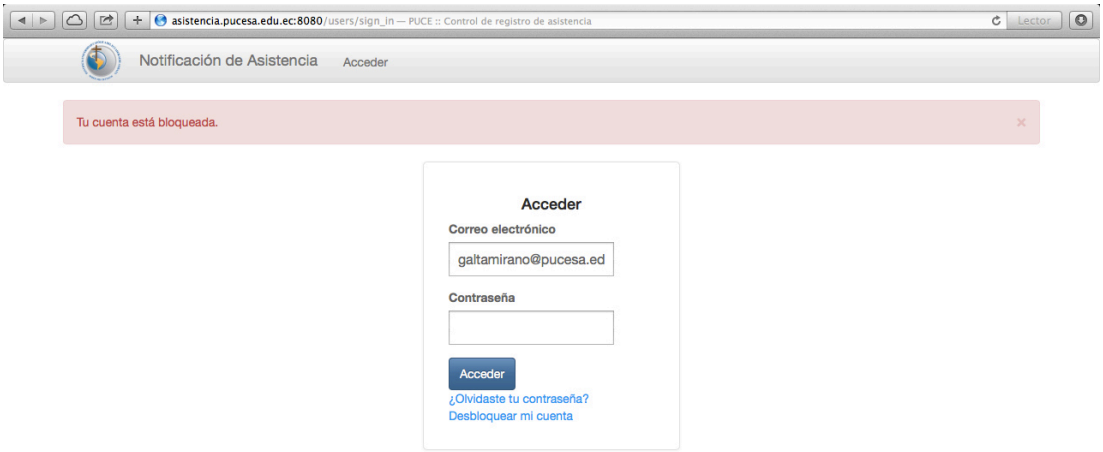


Ilustración 4.5: Bloqueo de Cuenta.

La Ilustración 4.6 muestra el proceso de desbloqueo de una cuenta al hacer clic en “Desbloquear mi cuenta”

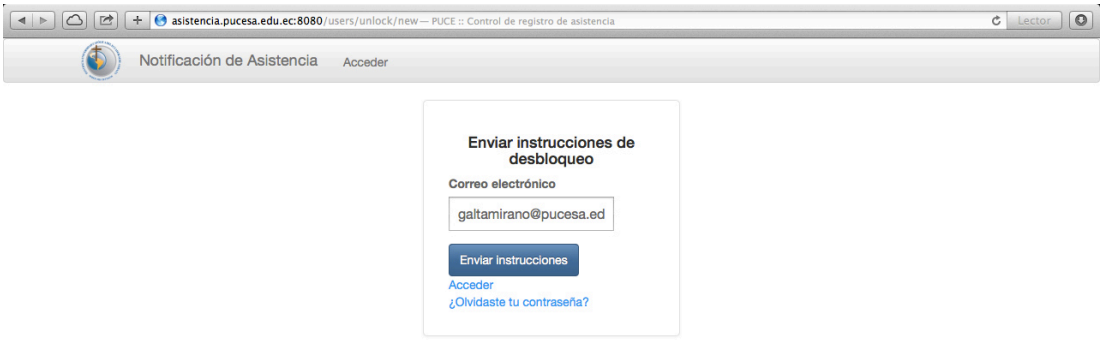


Ilustración 4.6: Desbloqueo de Cuenta.

La Ilustración 4.7 muestra la aplicación respondiendo ante un intento innecesario de desbloqueo de cuenta.

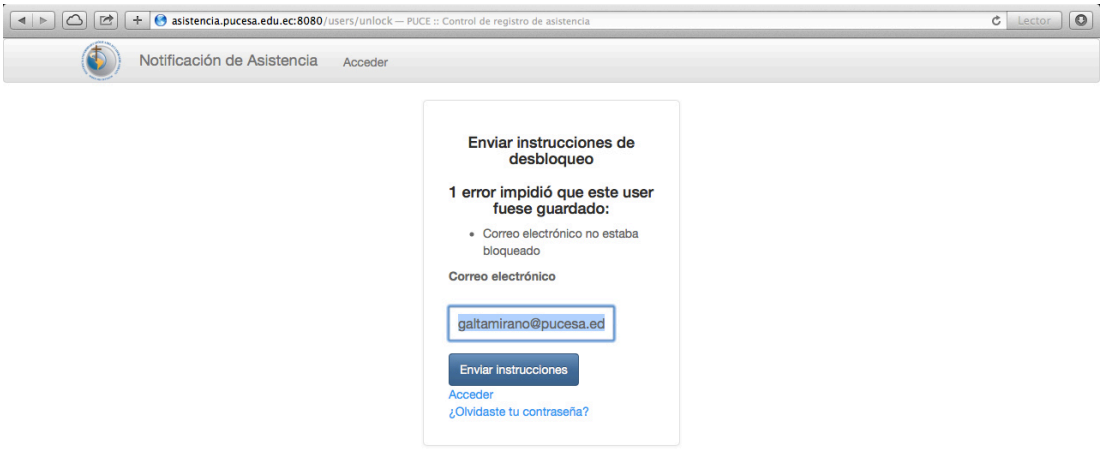


Ilustración 4.7: Desbloqueo de Cuenta No Bloqueada.

La Ilustración 4.8 muestra la confirmación de la aplicación sobre el inicio del proceso de desbloqueo de cuenta por medio de correo electrónico (véase Ilustración 4.9).

Ilustración 4.8: Envío de Instrucciones para Desbloquear Cuenta.

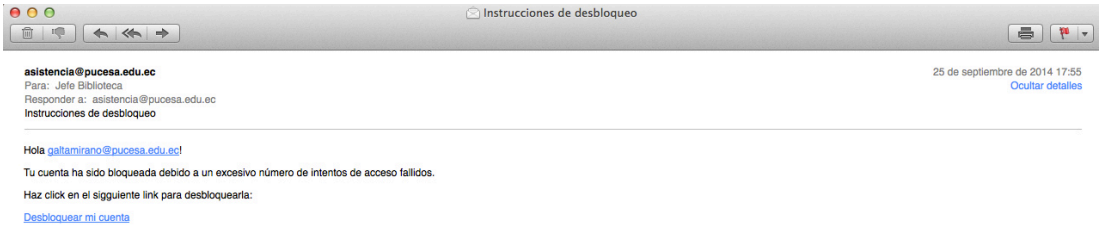


Ilustración 4.9: Instrucciones para Desbloquear Cuenta.

La Ilustración 4.10 muestra el caso en el que se intenta usar un “token” invalido.

Ilustración 4.10: Token Inválido para Desbloquear Cuenta.

La Ilustración 4.11 muestra el proceso de desbloqueo de cuenta exitosamente terminado, permitiendo ingresar nuevamente la información de acceso.

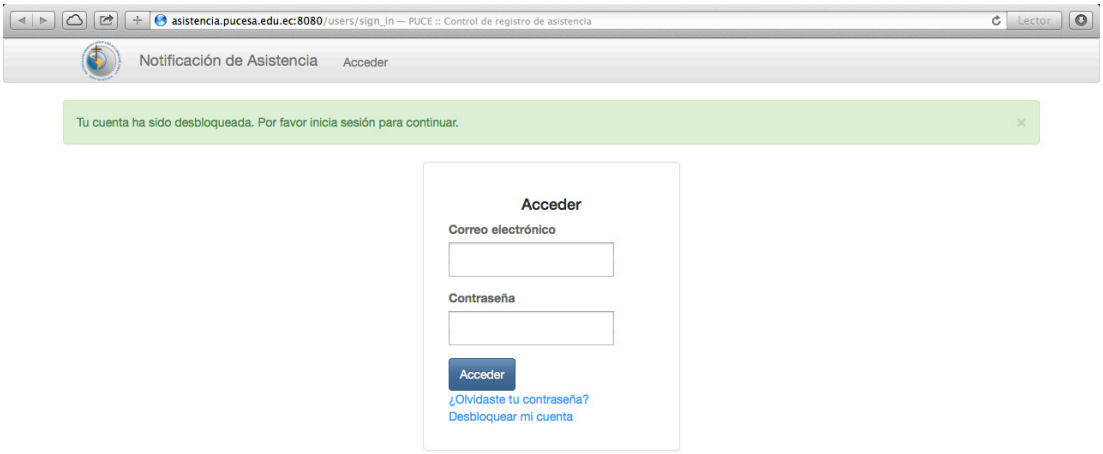


Ilustración 4.11: Desbloqueo de Cuenta.

La Ilustración 4.12 muestra el proceso de restablecimiento de contraseña al hacer clic en "Olvidé mi contraseña"

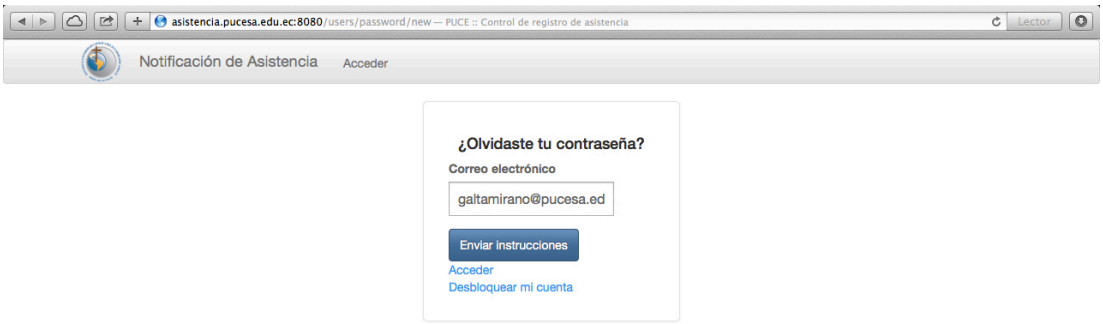


Ilustración 4.12: Restablecimiento de Contraseña

Las Ilustraciones 4.13, 4.14, 4.15 y 4.17 muestra un proceso exitoso de restablecimiento de contraseña. Mientras que la Ilustración 4.16 muestra el resultado del uso de un "token" inválido.

Ilustración 4.13: Envío de Instrucciones para Restablecimiento de Contraseña

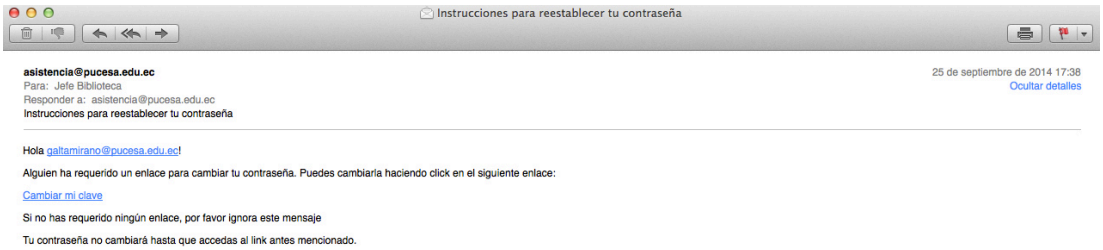


Ilustración 4.14: Instrucciones para Restablecimiento de Contraseña

Ilustración 4.15: Cambio de Contraseña

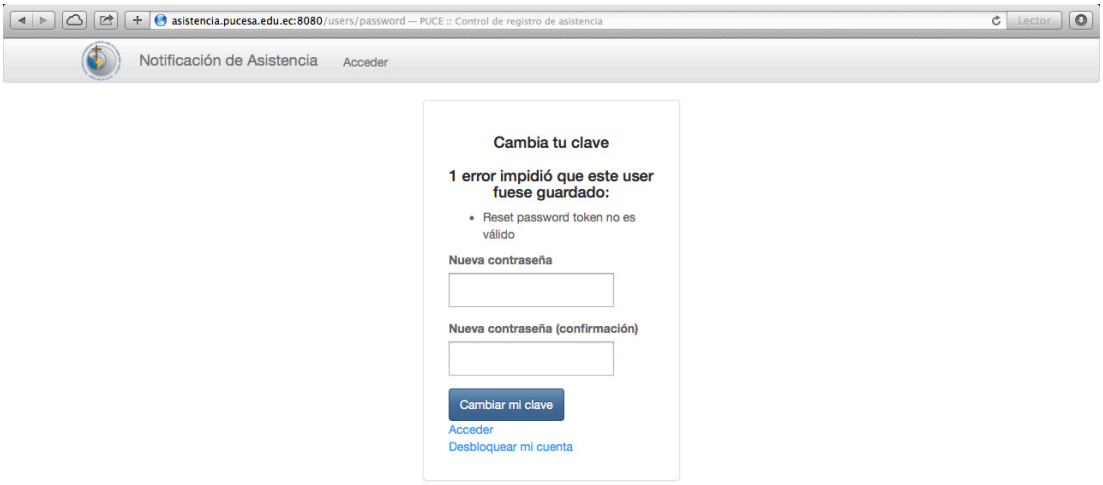


Ilustración 4.16: Token Inválido para Cambio de Contraseña

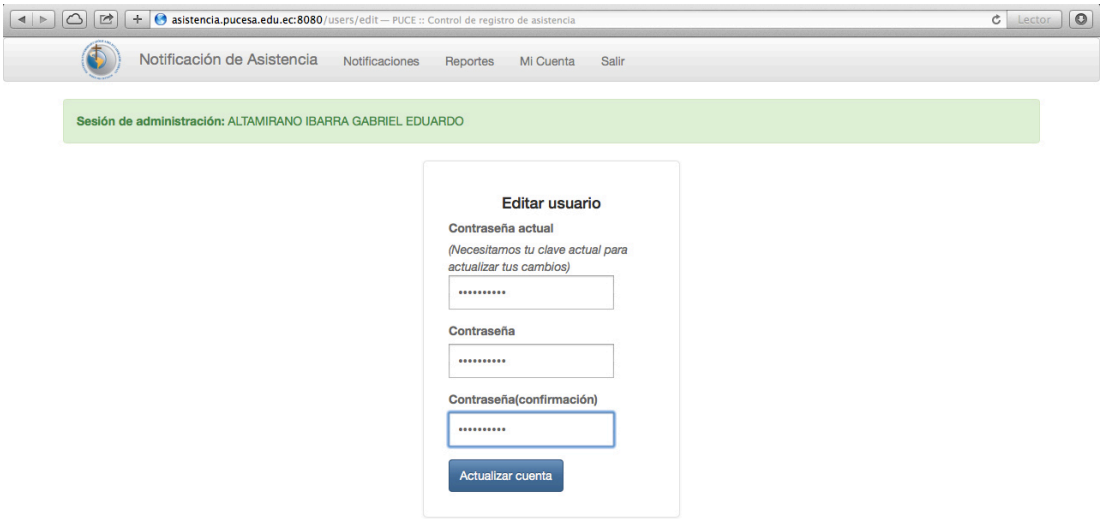


Ilustración 4.17: Cambio de Contraseña

4.1.2 Resultado de la Implementación de la Segunda Iteración

4.1.2.1 Resultado del Módulo "Notificaciones"

La Ilustración 4.18 muestra el módulo de notificaciones que en rol administrador gestiona información de todos

los trabajadores, caso contrario únicamente información personal del usuario.

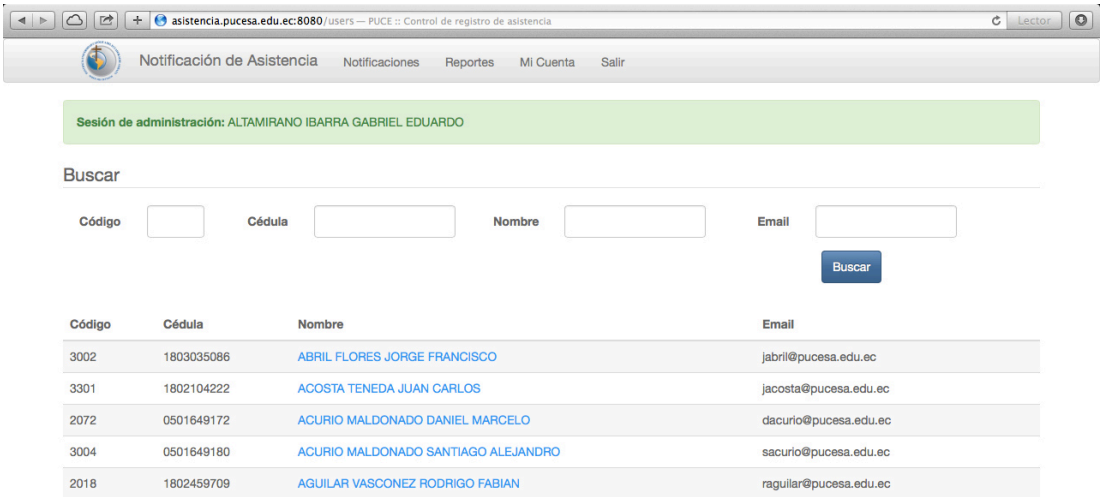


Ilustración 4.18: Módulo de "Notificaciones"

La Ilustración 4.19 evidencia la posibilidad de filtrar información mediante varios criterios para facilidad de navegación del administrador.

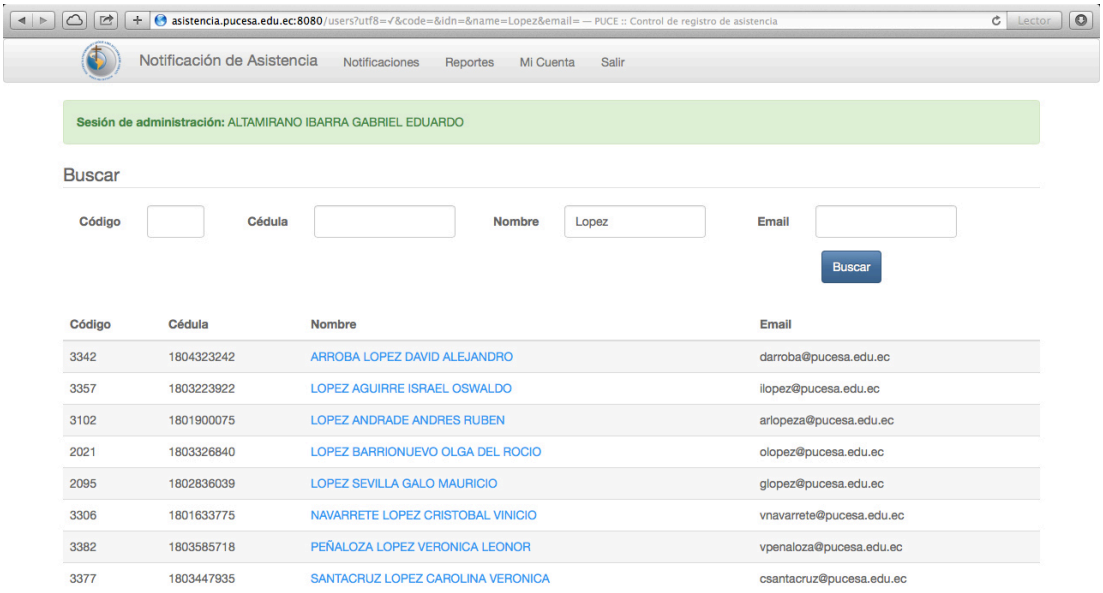


Ilustración 4.19: Filtros en "Notificaciones"

En la Ilustración 4.20 se muestra las notificaciones que por defecto son las del día anterior.

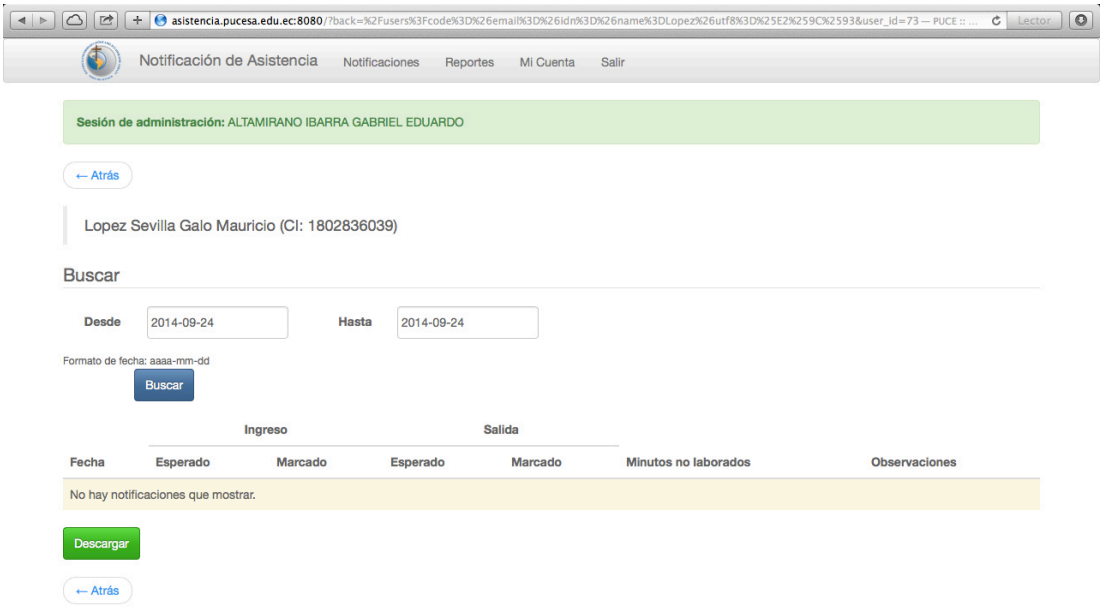


Ilustración 4.20: Ausencia de Notificaciones"

La Ilustración 4.21 demuestra cómo la aplicación permite parametrizar las fechas en las que se desea que se muestre las notificaciones generadas, mismas que pueden ser descargas en formato CSV (véase Ilustración 4.22) y fueron notificadas por correo electrónico institucional (véase Ilustración 4.23, 4.24, 4.25 y 4.26)

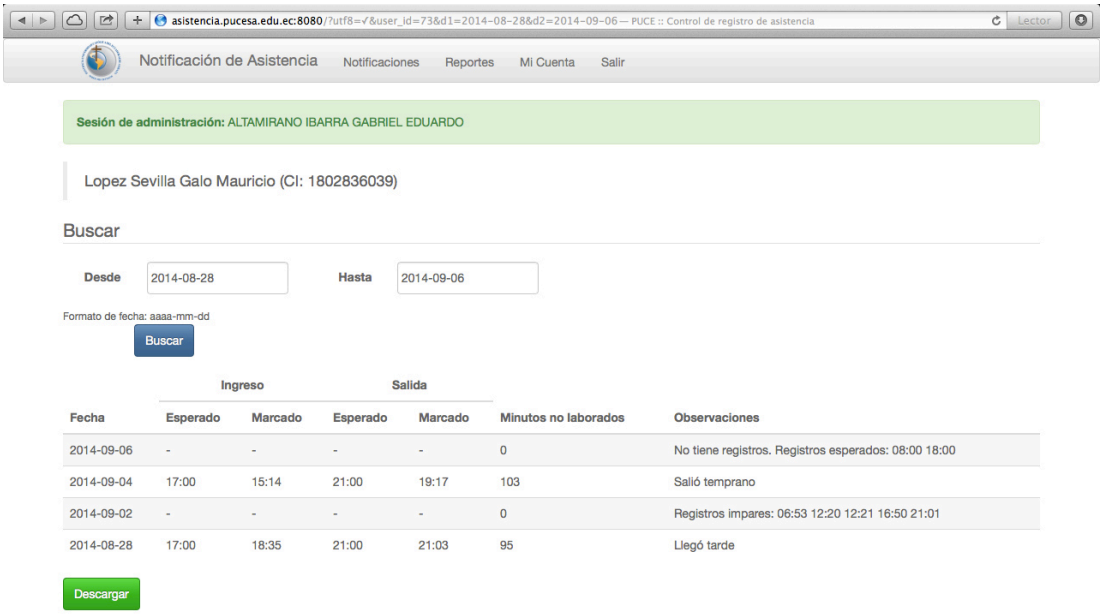


Ilustración 4.21: Historial de Notificaciones entre Fechas

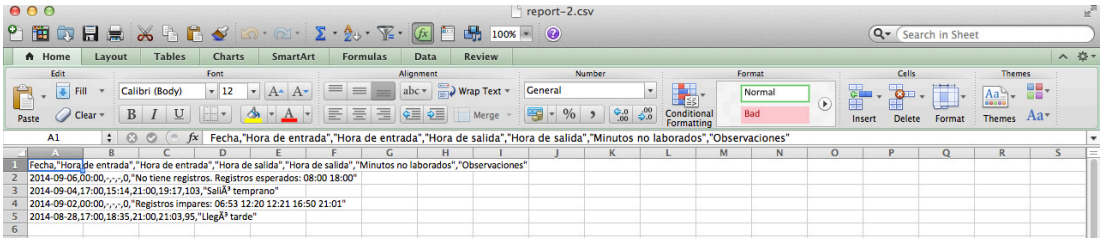


Ilustración 4.22: Descarga de Notificaciones en Formato (CSV).

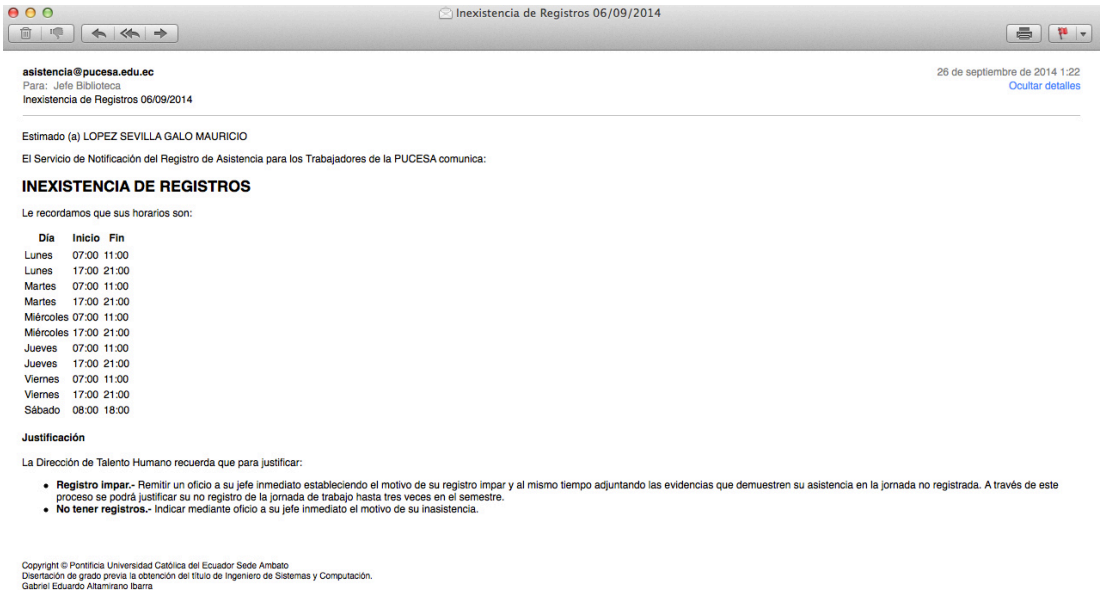
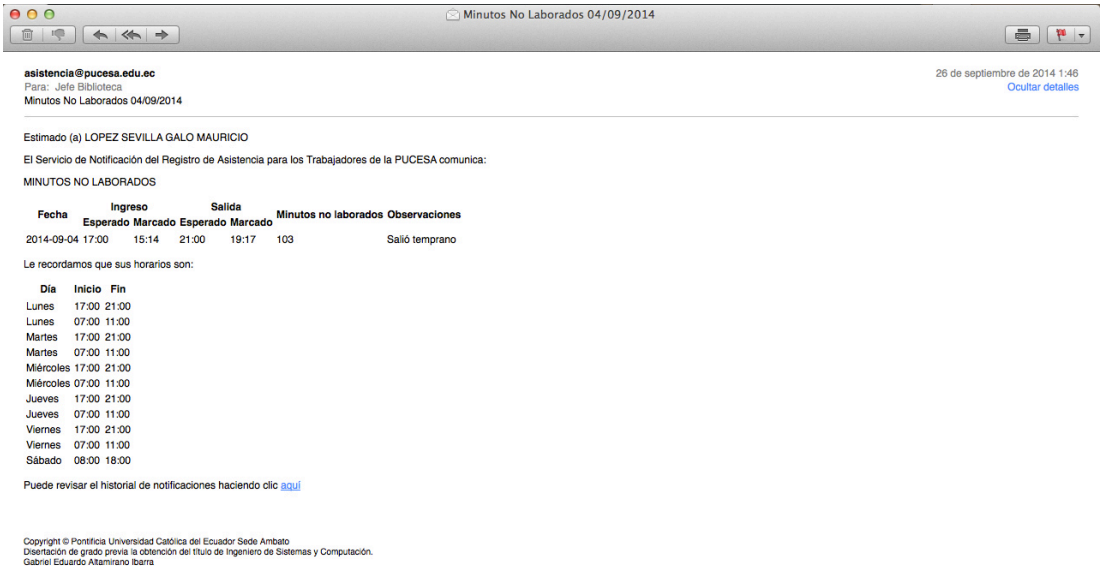


Ilustración 4.23: Correo Electrónico de Inexistencia de Registros.



Minutos No Laborados 04/09/2014

asistencia@pucesa.edu.ec
Para: Jefe Biblioteca
Minutos No Laborados 04/09/2014

26 de septiembre de 2014 1:46
Ocultar detalles

Estimado (a) LOPEZ SEVILLA GALO MAURICIO

El Servicio de Notificación del Registro de Asistencia para los Trabajadores de la PUCESA comunica:

MINUTOS NO LABORADOS

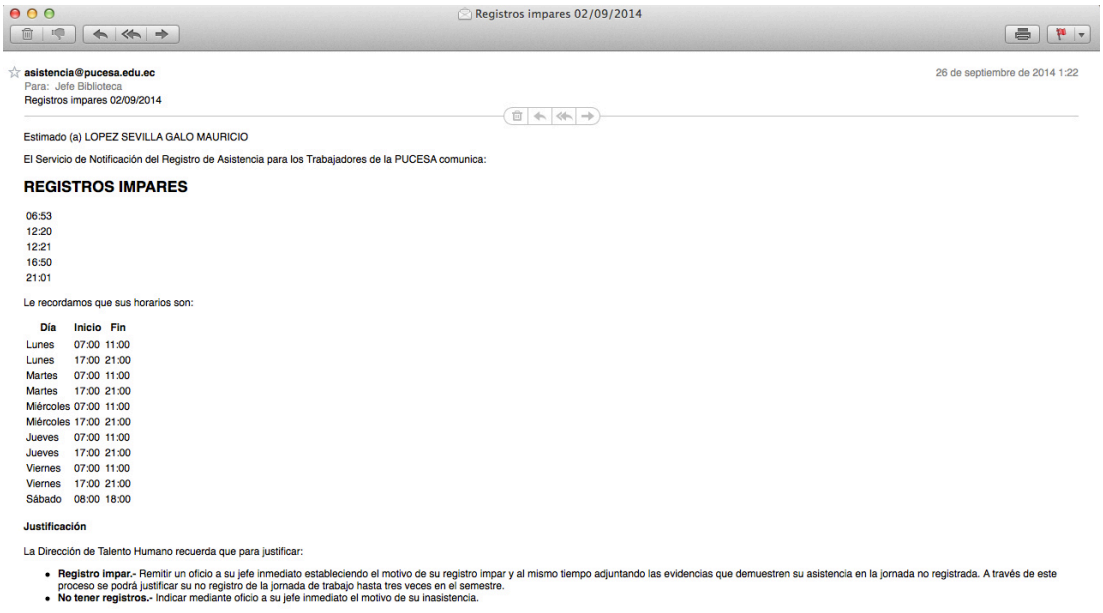
Fecha	Ingreso		Salida		Minutos no laborados	Observaciones
	Esperado	Marcado	Esperado	Marcado		
2014-09-04	17:00	15:14	21:00	19:17	103	Salió temprano

Le recordamos que sus horarios son:

Día	Inicio	Fin
Lunes	17:00	21:00
Lunes	07:00	11:00
Martes	17:00	21:00
Martes	07:00	11:00
Miércoles	17:00	21:00
Miércoles	07:00	11:00
Jueves	17:00	21:00
Jueves	07:00	11:00
Viernes	17:00	21:00
Viernes	07:00	11:00
Sábado	08:00	18:00

Puede revisar el historial de notificaciones haciendo clic aquí

Ilustración 4.24: Correo Electrónico de Minutos No Laborados (Temprano).



Registros impares 02/09/2014

asistencia@pucesa.edu.ec
Para: Jefe Biblioteca
Registros impares 02/09/2014

26 de septiembre de 2014 1:22

Estimado (a) LOPEZ SEVILLA GALO MAURICIO

El Servicio de Notificación del Registro de Asistencia para los Trabajadores de la PUCESA comunica:

REGISTROS IMPARES

06:53
12:20
12:21
16:50
21:01

Le recordamos que sus horarios son:

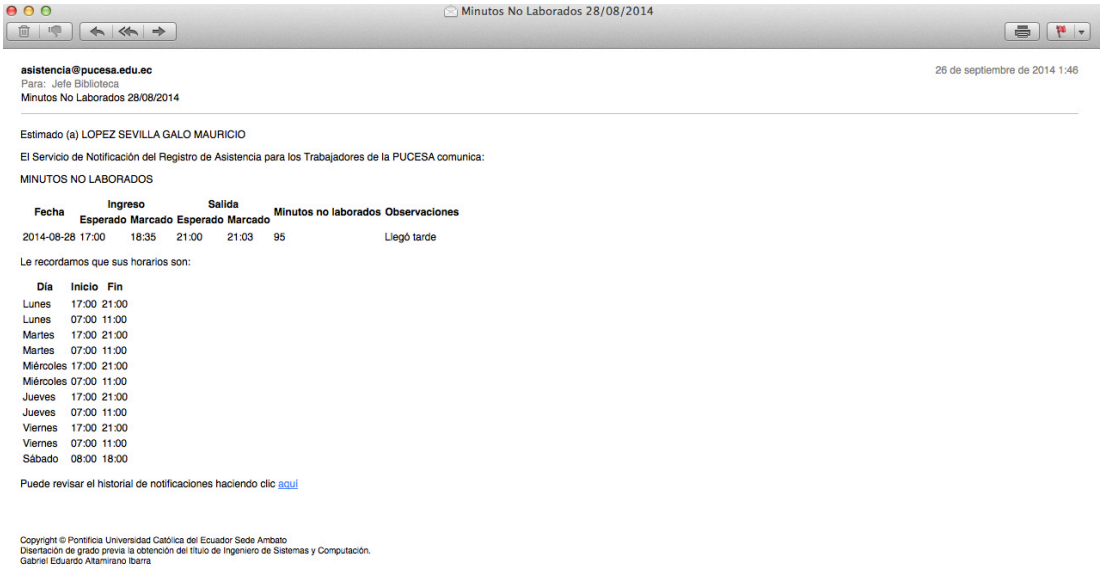
Día	Inicio	Fin
Lunes	07:00	11:00
Lunes	17:00	21:00
Martes	07:00	11:00
Martes	17:00	21:00
Miércoles	07:00	11:00
Miércoles	17:00	21:00
Jueves	07:00	11:00
Jueves	17:00	21:00
Viernes	07:00	11:00
Viernes	17:00	21:00
Sábado	08:00	18:00

Justificación

La Dirección de Talento Humano recuerda que para justificar:

- **Registro impar.-** Remitir un oficio a su jefe inmediato estableciendo el motivo de su registro impar y al mismo tiempo adjuntando las evidencias que demuestren su asistencia en la jornada no registrada. A través de este proceso se podrá justificar su no registro de la jornada de trabajo hasta tres veces en el semestre.
- **No tener registros.-** Indicar mediante oficio a su jefe inmediato el motivo de su inasistencia.

Ilustración 4.25: Correo Electrónico de Registros Impares.


Minutos No Laborados 28/08/2014

asistencia@pucesa.edu.ec 26 de septiembre de 2014 1:46
Para: Jefe Biblioteca
Minutos No Laborados 28/08/2014

Estimado (a) LOPEZ SEVILLA GALO MAURICIO

El Servicio de Notificación del Registro de Asistencia para los Trabajadores de la PUCESA comunica:

MINUTOS NO LABORADOS

Fecha	Ingreso		Salida		Minutos no laborados	Observaciones
	Esperado	Marcado	Esperado	Marcado		
2014-08-28	17:00	18:35	21:00	21:03	95	Llegó tarde

Le recordamos que sus horarios son:

Día	Inicio	Fin
Lunes	17:00	21:00
Lunes	07:00	11:00
Martes	17:00	21:00
Martes	07:00	11:00
Miércoles	17:00	21:00
Miércoles	07:00	11:00
Jueves	17:00	21:00
Jueves	07:00	11:00
Viernes	17:00	21:00
Viernes	07:00	11:00
Sábado	08:00	18:00

Puede revisar el historial de notificaciones haciendo clic aquí

Ilustración 4.26: Correo Electrónico de Minutos No Laborados (Tarde).

4.1.3 Resultado de la Implementación de la Tercera Iteración

4.1.3.1 Resultado del Módulo "Reportes"

La Ilustración 4.27 muestra al módulo de "Reportes" que es de uso exclusivo para el rol de administrador a través del cual se permite consultar las notificaciones de todos los trabajadores entre dos fechas, el reporte es exportable a formato CSV.

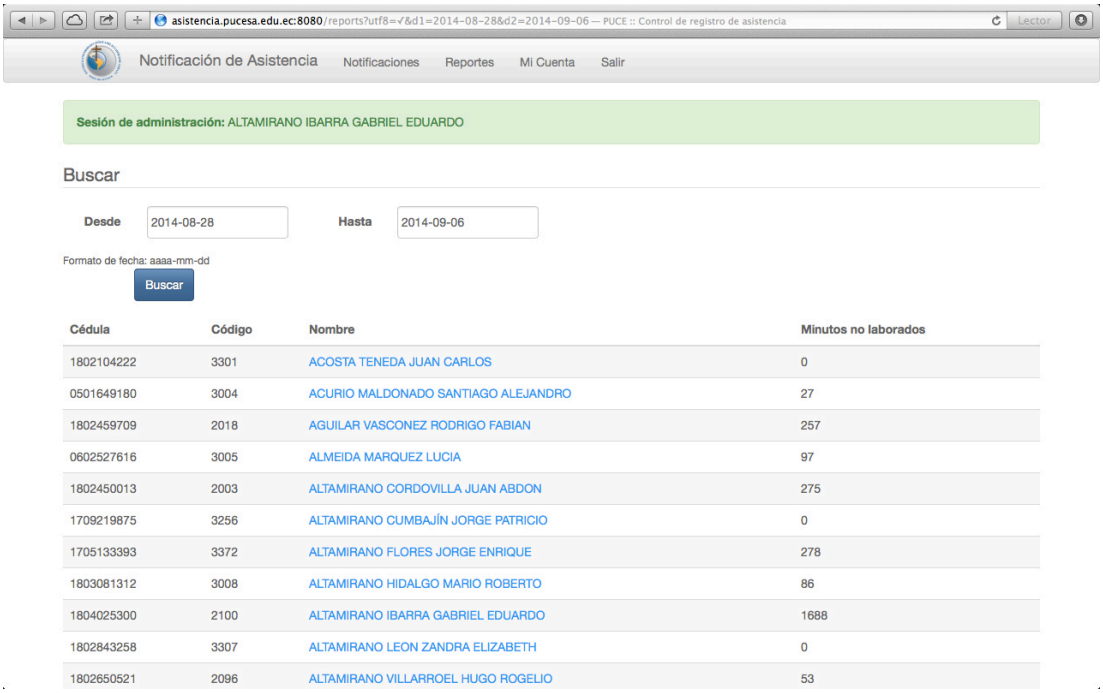


Ilustración 4.27: Módulo de "Reportes".

4.2 Validación de Resultados

Ambato, 28 de Noviembre del 2014

Ingeniero
Galo Mauricio López Sevilla
DIRECTOR DE LA ESCUELA DE INGENIERÍA DE SISTEMAS DE LA PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE AMBATO
Presente.-

De mi consideración:

Junto con saludarlo, le comunico que la disertación de Gabriel Eduardo Altamirano Ibarra con tema "IMPLEMENTACIÓN DE UN SERVICIO DE NOTIFICACIÓN DEL REGISTRO DE ASISTENCIA PARA LOS TRABAJADORES DE LA PUCESA", ha sido revisada, por lo que valido este trabajo.

Por su atención quedo de usted muy agradecido.

Atentamente,

Ing. David Camacho Pedroza
DIRECTOR DE TALENTO HUMANO PUCESA

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR SEDE AMBATO
DIRECCION DE TALENTO HUMANO

DIRECCIÓN TALENTO HUMANO
Av. Manuelita Sáenz s/n
sector El Tropezón
Telf.: 593 3 2586 008
ext. 110
www.pucesa.edu.ec

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. El framework Ruby on Rails propende al desarrollo ágil gracias a su arquitectura Modelo Vista Controlador (MVC) por su libre configuración en el controlador que a su vez da lógica a las vistas y un sencillo manejo de la base de datos a través de Active Record.

2. TorqueBox es una plataforma de aplicaciones Ruby que soporta la tecnología de Ruby on Rails, edificados sobre el servidor de aplicaciones Java JBoss y lenguaje de programación para JVM JRuby, ofrece funciones tales como clustering, balanceo de carga y alta disponibilidad.

3. El IDE RubyMine de JetBrains posee capacidades inteligentes como asistencia de codificación, refactorización inteligente de código y análisis de

código. Fácil configuración de proyectos, gestión automática de Ruby Gems, apoyo Rake.

4. Extreme Programming como metodología de desarrollo entrega software flexible, rápido y real con las necesidades planteadas por el usuario debido a sus ciclos cortos de desarrollo y la estrecha relación que posee con el usuario.

RECOMENDACIONES

1. Estandarizar políticas institucionales de registro ante la implementación de este servicio para garantizar su total funcionalidad.

2. Usar Jruby en soluciones Ruby on Rails que involucren tecnología Java por facilidad de integración.

3. Ser simple en el diseño del software desarrollando únicamente lo que se necesita, muy comunicativo y respetuoso con las personas ya que Extreme

Programming es una metodología enfocada en las personas.

4. Desarrollar programas sencillos para explorar soluciones que reduzcan el riesgo de problemas técnicos o de diseño y aumenten la fiabilidad en la estimación de una historia de usuario.

5. Considerar para la estimación del tiempo de desarrollo las siguientes equivalencias: una semana es un punto de estimación, una semana son 5 días laborables, el día laborable es de 5 horas y un equipo de desarrollo de dos personas.

6. Construir pruebas de unidad legibles que dirijan y validen el desarrollo permitiendo una ágil retroalimentación en el diseño y la corrección de errores.

7. Las pruebas de aceptación deben ser diseñadas entre el usuario y el equipo de desarrollo ya que supervisan que una historia de usuario sea implementada de acuerdo a las necesidades planteadas por el usuario.

BIBLIOGRAFÍA

1. Amano Cincinnati. (2002). Hand Punch-32: User's Guide.

2. Beck, K. (2000). Extreme programming eXplained : embrace change. Reading, MA : Addison-Wesley.

3. Beck, K., & Fowler, M. (2001). Planning extreme programming. Boston: Addison-Wesley.

4. Edelson, J., & Liu, H. (2008). JRuby Cookbook. O''Reilly Media.

5. Herrera Ríos, E. (2011). Arrancar con HTML5: curso de programación. México: Alfaomega Grupo Editor.

6. Jain, A. K., Flynn, P., & Ross, A. A. (2008). Handbook of Biometrics. New York, Estados Unidos de América: Springer.

7. Jeffries, R. (2000). Extreme programming installed. Boston: Addison-Wesley.

8. Marchioni, F. (2013). JBoss AS 7 Development. Olton, Birmingham, GBR : Packt Publishing.

9. O' Gorman, L. (2003). Comparing Passwords, Token, and Biometrics for User Authentication. Proceedings of the IEEE , 12, 2019-2040.

10. Pressman, R. S., Campos, O. V., & Enríquez, B. J. (2010). Ingeniería del software: Un enfoque práctico. México: McGraw-Hill.

11. Rajshekhar, A. (2008). Building Dynamic Web 2.0 Websites with Ruby on Rails. Birmingham, Reino Unido: Packt Publishing Ltd.

12. Recognition Systems. (2003). HandPunch 2000: Manual. Recognition Systems, Inc.

13. Smith, J. (2001). A comparison of RUP and XP. ALM Company.

14. SquareNet Software. (2008). SquareNet Software. Retrieved Febrero 2014, from Home: http://www.squarenet.com.ec/index.html

15. SquareNet Software. (2008). SquareNet Software. Retrieved Febrero 2010, from Control de Asistencia: http://www.squarenet.com.ec/moduloca.html

16. SquareNet Software. (2008). SquareNet Software. Retrieved Febrero 2010, from Plataforma SquareNet: http://www.squarenet.com.ec/plataforma.html

17. Tibbon Technology. (2009). DS100 Serial Device Server.

18. Wells, D. (1999). Extreme Programming Agile Process. Retrieved Marzo 23, 2014, from Extreme Programming: http://www.extremeprogramming.org/

ANEXOS

Anexo 1: Manual de Usuario

El usuario debe ingresar a la siguiente dirección: http://asistencia.pucesa.edu.ec:8080

Acceso a la Aplicación Web.

Para iniciar sesión el usuario debe ingresar su correo electrónico institucional (xxxxx@pucesa.edu.ec) y la contraseña (número de cédula).

Ilustración: Acceso a la Aplicación Web.

Desbloqueo de Cuenta

Debido a intentos fallidos la cuenta puede bloquearse, hacer clic entonces en "Desbloquear mi cuenta"

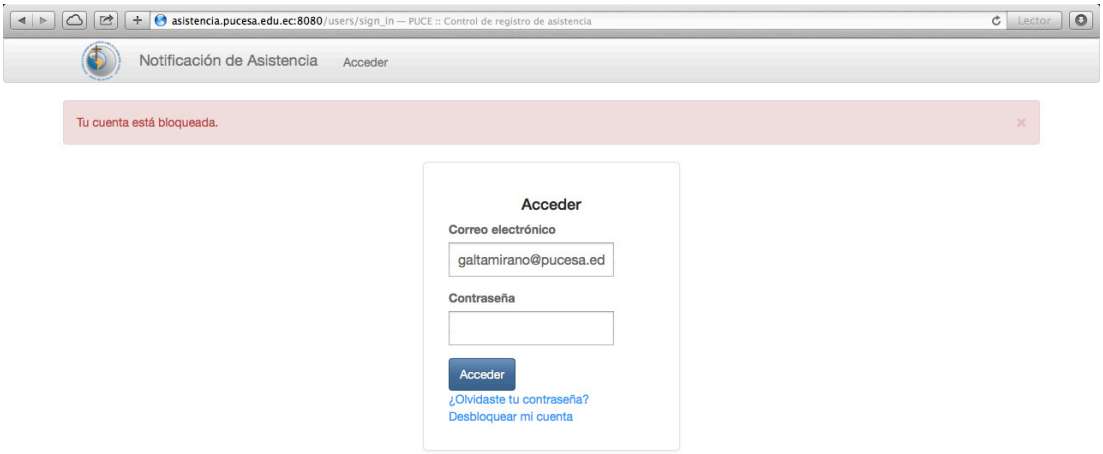


Ilustración: Bloqueo de cuenta.

Ingresar el correo electrónico institucional y hacer clic en "Enviar instrucciones"

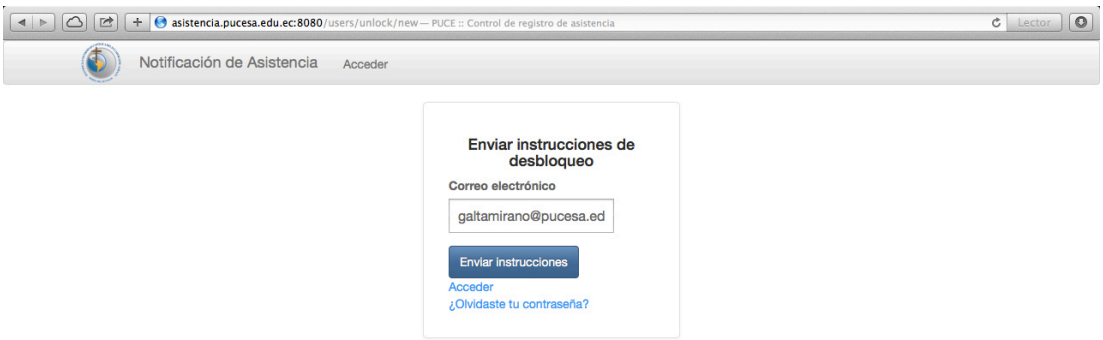


Ilustración: Desbloqueo de cuenta.

Revisar la bandeja de entrada del correo electrónico institucional y seguir las instrucciones de desbloqueo.

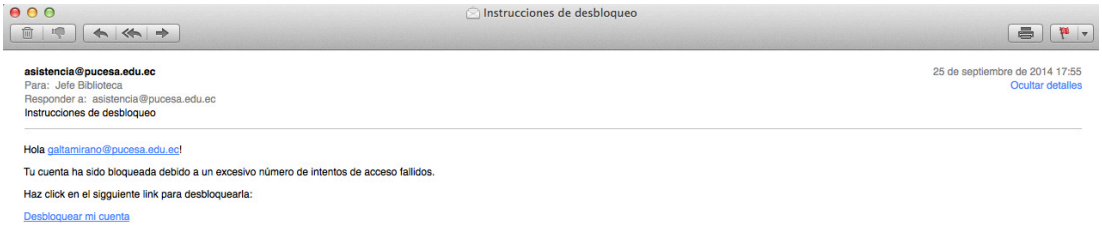


Ilustración: Instrucciones de desbloqueo.

Restablecimiento de Contraseña

Si el usuario olvidó su contraseña debe hacer clic en "¿Olvidaste tu contraseña?", ingresar el correo electrónico institucional y clic en "Enviar instrucciones"

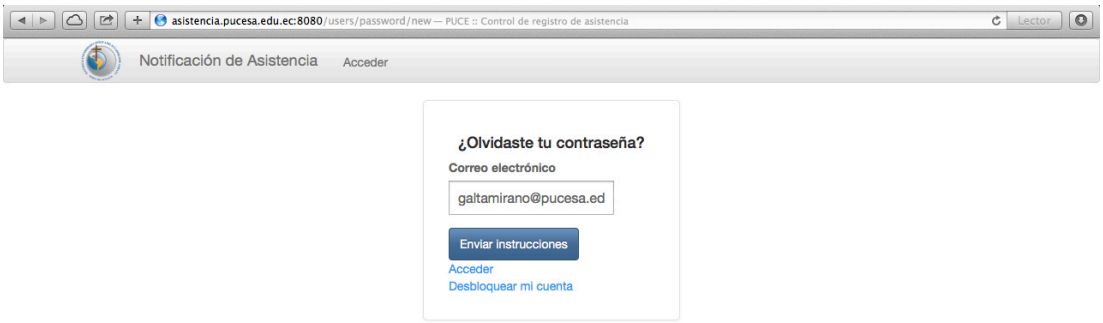


Ilustración: Restablecimiento de Contraseña

Revisar la bandeja de entrada del correo electrónico institucional y seguir las instrucciones para restablecer la contraseña.

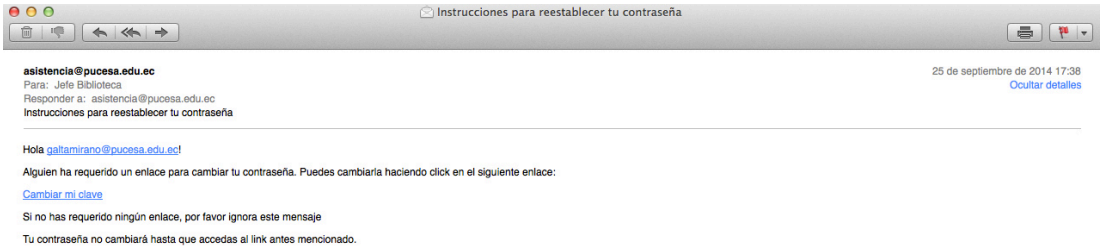


Ilustración: Instrucciones para Restablecimiento de Contraseña

El usuario debe ingresar la nueva contraseña y luego confirmarla.

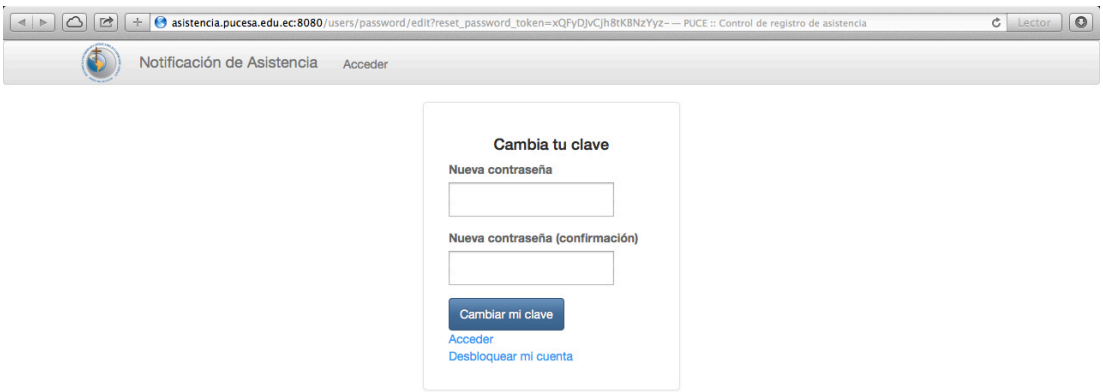


Ilustración: Cambio de Contraseña

Aplicación

Una vez que el usuario haya ingresado a la página principal del sistema puede informarse sobre su descripción, módulos y justificación.

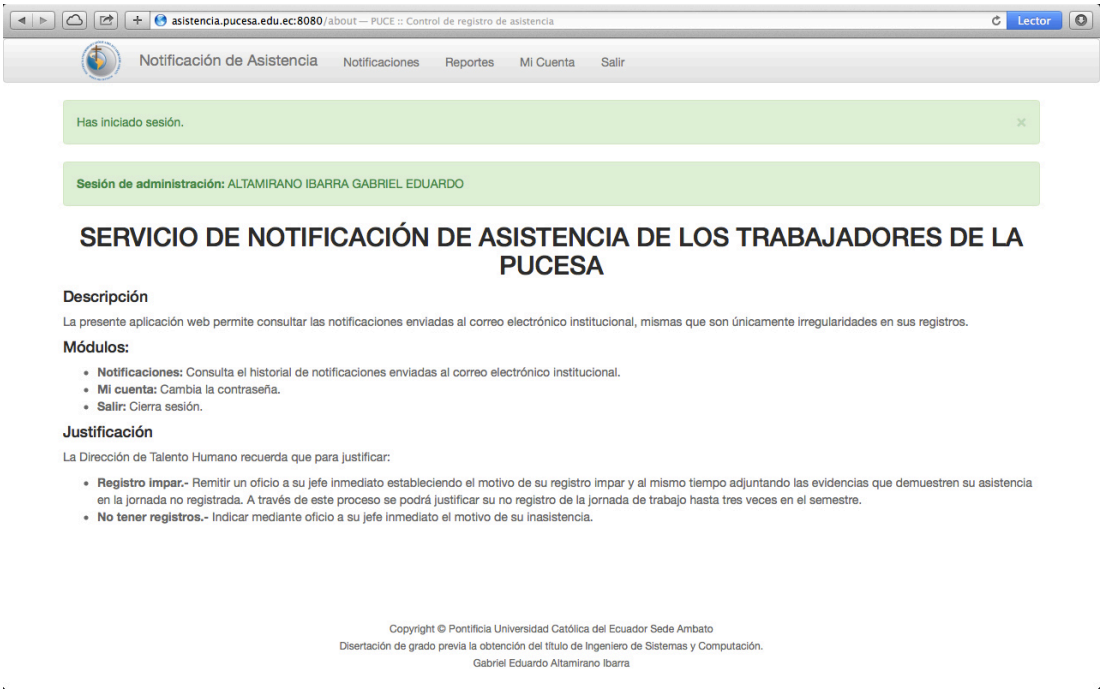


Ilustración: Página Principal

Notificaciones

El módulo de notificaciones muestra las novedades que el usuario posee en un rango de fechas o del día anterior por defecto. El administrador tendrá acceso a datos de todos los empleados pudiendo ágilmente buscarlos por varios criterios.

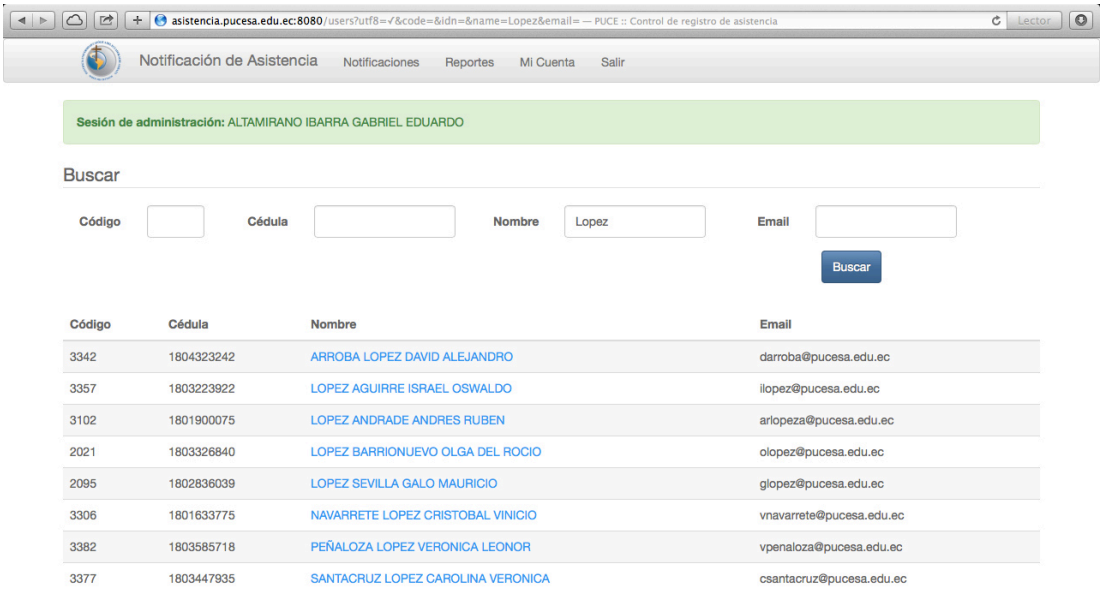


Ilustración: Filtros en "Notificaciones"

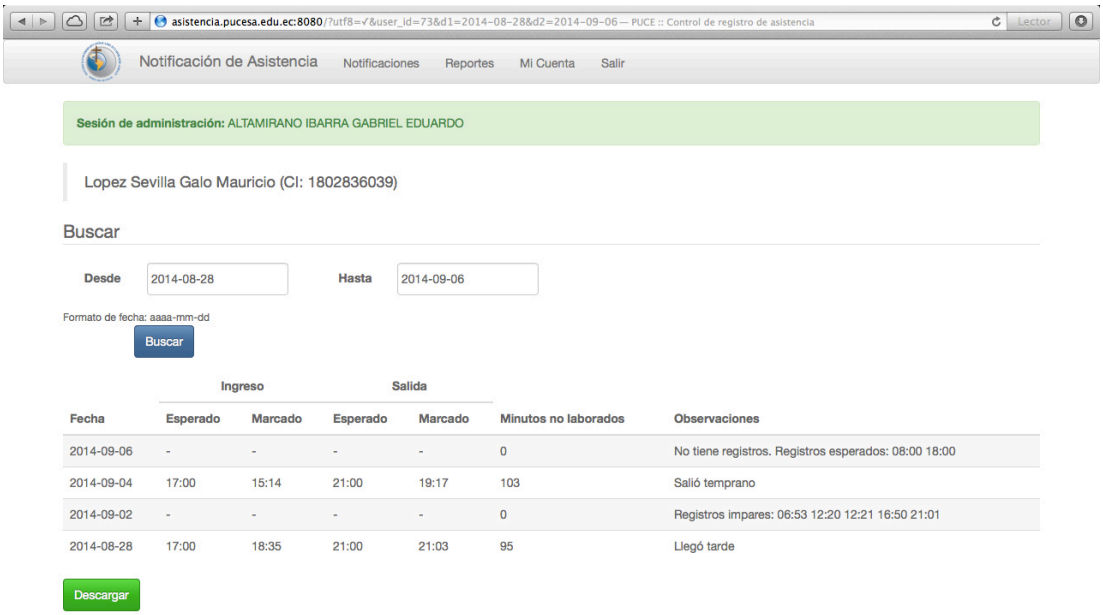


Ilustración: Historial de Notificaciones entre Fechas

El botón "Descargar" permite obtener la información en un archivo separado por comas (CSV).

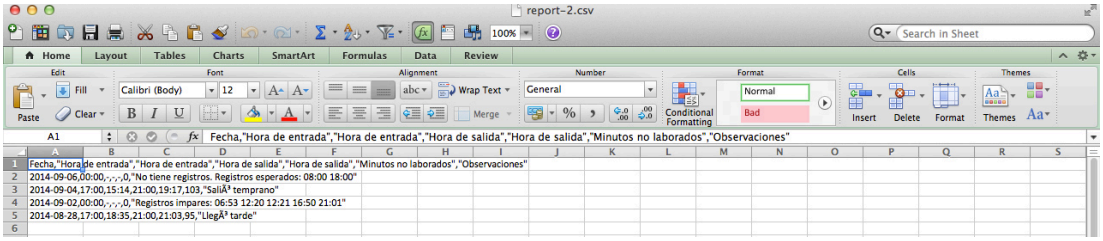


Ilustración: Descarga de Notificaciones en Formato (CSV).

Reportes

El módulo de "Reportes" es de uso exclusivo para los usuarios con privilegios de administrador.

Consiste en extraer las notificaciones de asistencia de todos los empleados en un rango de fechas, permitiendo también descargarlo en formato CSV.

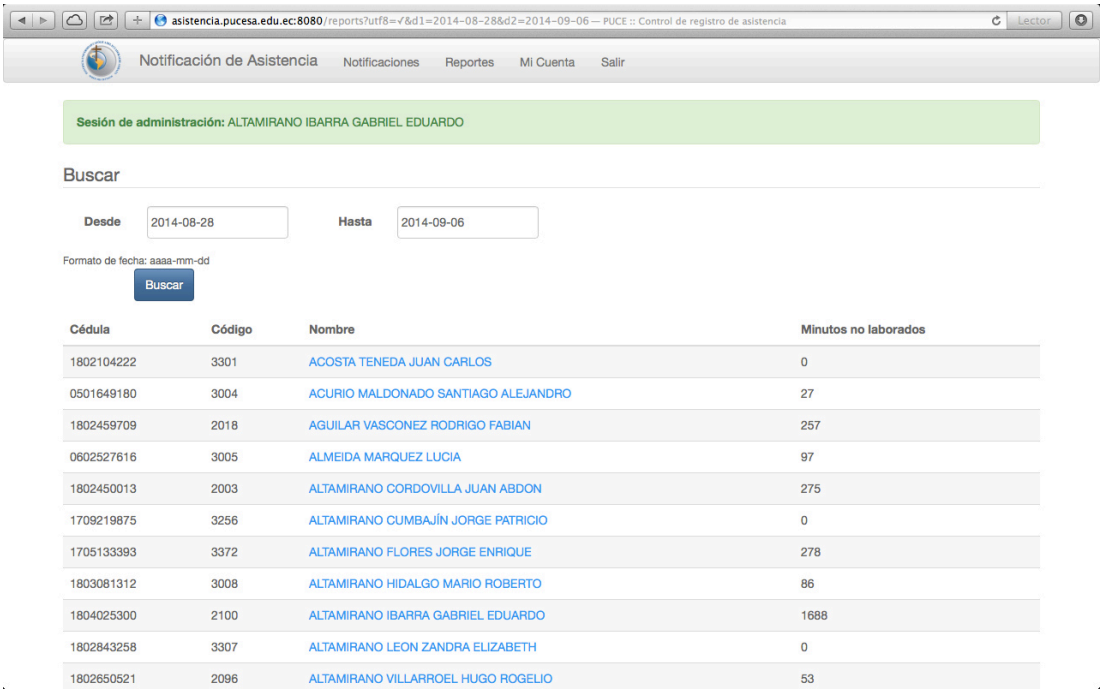


Ilustración: Correo Electrónico de Minutos No Laborados (Tarde).

Cuenta

Finalmente el módulo “Cuenta” permite a los usuarios cambiar la contraseña de acceso.

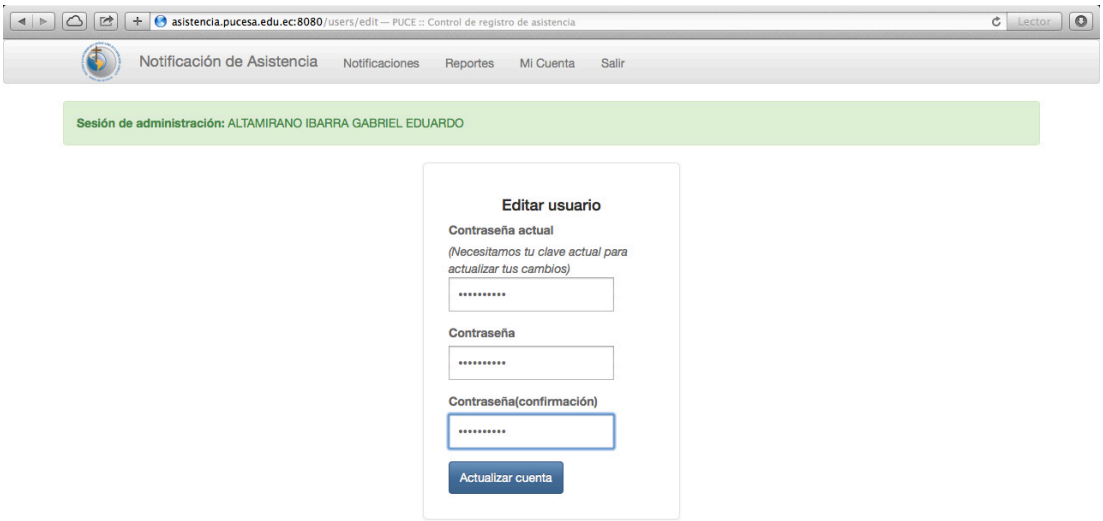


Ilustración: Correo Electrónico de Minutos No Laborados (Tarde).