

Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

INGENIERÍA DEL SOFTWARE I

**SISTEMA DE GESTIÓN VETERINARIA**

**Profesores**:

* Alejandra Matoso.
* Medina María José.
* Oscar Adolfo Vallejos.

**Integrantes**:

* Abrahan Ramirez, Ulises Alejandro – LU: 49429
* Alia, Lucas – LU: 53181
* Collar, Tomás – LU: 52508

**1 SISTEMA DE GESTIÓN VETERINARIA**

**1.1 Breve historia del arte:**

Teniendo en cuenta el rápido avance de la tecnología y el crecimiento del uso que la acompaña, es necesario tener en cuenta que una PYME no puede quedarse al margen de esta situación.

En vista de eso, creemos que es necesario implementar un sistema de digitalización para la misma, con el fin de innovar en nuevos aspectos, y generar un nuevo tipo de demanda.

**1.2 Objetivos/Propósitos:**

 Establecimos como objetivo la creación de un sistema de Gestión que permitirá brindar distintos servicios, tales como: reserva de turnos, registro de pacientes, archivado de historias clínicas y automatización de stock. La audiencia a quien va dirigido es tanto como al personal del consultorio, ya sean secretarios o profesionales y al usuario final.

**1.3 Alcance**

El sistema a desarrollarse de denotará por el nombre SAV.

Este sistema permitirá mostrar la lista de pacientes (en este caso los animales) correspondiente a la fecha solicitada para brindar información al Profesional de Salud que se encuentre trabajando y llevar un registro de los pacientes. Con el fin poder Automatizar procesos o tareas al Personal encargado, poder ahorrar tiempo, aumentar la productividad y obtener como resultado un beneficio importante del cual es: eliminar o bien disminuir en una gran tamaño el papeleo. Lo que se espera a futuro es poder mejorar la experiencia del cliente (Los respectivos dueños de cada animal) para solucionar de una manera más satisfactoria el servicio que se le ofrece.

El sistema le permitirá a la empresa, poder mejorar en sus decisiones, ya que el sistema sabrá cuándo comprar y cuando, no analizando su demanda, esto lo logrará mediante la utilización del software adecuado.

**1.4 Definiciones, siglas y abreviaturas**

*RF* → Requisitos Funcionales

*RNF →* Requisitos NO Funcionales

SAV → Sistema Autogestión de Veterinaria

IF → Información

IEEE → Instituto de Ingeniería Eléctrica y Electrónica

**1.5 Referencias**

|  |  |  |
| --- | --- | --- |
| **Referencia** | **Titulo** | **Ruta** |
| 1 | IEEE830 - 1998 | <https://www.ctr.unican.es/asignaturas/is1/IEEE830_esp.pdf> |
| 2 | Manual de Procedimientos | Documentos de la Organización |
| 3 | Organigrama | Documentos de la Organización |

**2 Descripción Global**

**2.1 Perspectiva del producto**

El Sistema de Gestión a llevarse a cabo es Modular mente Independiente, esto quiere decir que el mismo no tendrá relaciones con otros Sistema ni formará parte de otro sub-sistema.

**2.2 Funciones del producto**

→ El Sistema permite Registrar turno.

→ Listar la lista de Pacientes con Turnos.

→ El Sistema puede Registrar la situación(diagnóstico) del Paciente al terminar de ser atendido.

→ El Sistema ofrece la opción de dar de Baja un turno solicitado anteriormente.

→ El Sistema actualizará el Stock Automáticamente

**2.3Características del Usuario**

|  |  |  |
| --- | --- | --- |
| *Tipo de usuario* | *Formación (nivel educativo)* | *Experiencia y Especialización técnica* |
| *Medico Veterinario* | *Educación Superior* | *Conocimientos Básicos sobre Operador de Computadoras* |
| *Nutricionista Animal* | *Educación Superior* | *Conocimientos Básicos sobre Operador de Computadoras* |
| *Anestesiólogo Veterinario* | *Educación Superior* | *Conocimientos Básicos sobre Operador de Computadoras* |
| *Cirujano* | *Educación Superior* | *Conocimientos Básicos sobre Operador de Computadoras* |
| *Oficinista* | *Educación Básica*  *(Secundario Completo)* | *Conocimientos Básicos sobre Operador de Computadoras* |

**2.4 Restricciones**

→ El Lenguaje a Utilizar para el desarrollo es Java.

→ La especificación de requerimientos de software se deberá realizar según la norma IEEE   830-1998.

**2.5Atención y Dependencias**

Este Sistema no cuenta con relación a terceros por lo tanto funciona independientemente sin ningún tipo de Dependencias.

**3 Requisitos específicos**

**3.1 Requisitos funcionales**

|  |  |
| --- | --- |
| *Nº de requisito* | *Descripción del requisito* |
| *RF#01* | *El sistema debe permitir a los usuarios solicitar turnos a través del sistema según la especialidad que necesite.* |
| *RF#02* | *El Sistema permitirá listar todos los turnos registrados por fecha solicitada y Especialidad* |
| *RF#03* | *El Sistema permitirá al usuario Ingresar al Sistema por Usuario y Contraseña Generado por el Administrador* |
| *RF#04* | *El Sistema permitirá la cancelación, después de registrar el turno en un tiempo menor o igual a 24 horas hábiles.* |
| *RF#05* | *El Sistema permitirá al Profesional de Salud Indicar el Diagnóstico del Paciente según Clasificación y legajo del cliente* |
| *RF#06* | *El Sistema permitirá Actualizar el Stock Automáticamente Consultando la Base de datos con un Algoritmo Matemático* |
| *RF#07* | *El Sistema Imprimirá un mensaje en Pantalla avisando al usuario final cuando El Stock sea Actualizado* |

**3.2 Requisitos no funcionales**

|  |  |  |
| --- | --- | --- |
| *Nº de requisito* | *Descripción del requisito* | *Clasificación - Requerimiento* |
| *RNF#01* | *El Sistema se implementará en el lenguaje de Programación es Java* | Organizacionales/Implementación |
| *RNF#02* | *El tiempo de respuesta al usuario deberá realizarse en un lapso de 10 a 20 segundos* | Producto/Eficiencia/Rendimiento |
| *RNF#03* | *La especificación de requerimientos de software se deberá realizar según la norma IEEE 830-1998.* | Organizacionales/Estándares |
| *RNF#04* | *El Sistema Asignará automáticamente un número único de Legajo a cada cliente* | Organizacionales/Implementación |
| *RNF#05* | *El diseño y Arquitectura se debe producir con la herramienta CASE: Umbrello* | Organizacionales/Implementación |
|  |  |  |

**4 Metodología : Programación Extrema (xp)**

**4.1 Planificación del Proyecto**

....... .......Historias de usuario: El primer paso de cualquier proyecto que siga la metodología X.P es definir las historias de usuario con el cliente. Las historias de usuario tienen la misma finalidad que los casos de uso pero con algunas diferencias: Constan de 3 ó 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados, etc. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia. El tiempo de desarrollo ideal para una historia de usuario es entre 1 y 3 semanas.

.......**Release planning:** .Después de tener ya definidas las historias de usuario es necesario crear un plan de publicaciones, en inglés "Release plan", donde se indiquen las historias de usuario que se crearán para cada versión del programa y las fechas en las que se publicarán estas versiones. Un "Release plan" es una planificación donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias que serán implementadas en cada versión del programa. Después de un "Release plan" tienen que estar claros estos cuatro factores: los objetivos que se deben cumplir (que son principalmente las historias que se deben desarrollar en cada versión), el tiempo que tardarán en desarrollarse y publicarse las versiones del programa, el número de personas que trabajarán en el desarrollo y cómo se evaluará la calidad del trabajo realizado.

....... **Iteraciones** Todo proyecto que siga la metodología X.P. se ha de dividir en iteraciones de aproximadamente 3 semanas de duración. Al comienzo de cada iteración los clientes deben seleccionar las historias de usuario definidas en el "Release planning" que serán implementadas. También se seleccionan las historias de usuario que no pasaron el test de aceptación que se realizó al terminar la iteración anterior. Estas historias de usuario son divididas en tareas de entre 1 y 3 días de duración que se asignarán a los programadores.

....... **Velocidad del proyecto:** La velocidad del proyecto es una medida que representa la rapidez con la que se desarrolla el proyecto; estimarla es muy sencillo, basta con contar el número de historias de usuario que se pueden implementar en una iteración; de esta forma, se sabrá el cupo de historias que se pueden desarrollar en las distintas iteraciones. Usando la velocidad del proyecto controlaremos que todas las tareas se puedan desarrollar en el tiempo del que dispone la iteración. Es conveniente reevaluar esta medida cada 3 ó 4 iteraciones y si se aprecia que no es adecuada hay que negociar con el cliente un nuevo "Release Plan".

....... **Programación en pareja:** La metodología X.P. aconseja la programación en parejas pues incrementa la productividad y la calidad del software desarrollado. El trabajo en pareja involucra a dos programadores trabajando en el mismo equipo; mientras uno codifica haciendo hincapié en la calidad de la función o método que está implementando, el otro analiza si ese método o función es adecuado y está bien diseñado. De esta forma se consigue un código y diseño con gran calidad.

....... **Reuniones diarias.** Es necesario que los desarrolladores se reúnan diariamente y expongan sus problemas, soluciones e ideas de forma conjunta. Las reuniones tienen que ser fluidas y todo el mundo tiene que tener voz y voto.

**4.2 Diseño**

....... **Diseños simples:** La metodología X.P sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible e impleméntable que a la larga costará menos tiempo y esfuerzo desarrollar.

....... **Glosarios de términos:** Usar glosarios de términos y un correcta especificación de los nombres de métodos y clases ayudará a comprender el diseño y facilitará sus posteriores ampliaciones y la reutilización del código.

....... **Riesgos:** Si surgen problemas potenciales durante el diseño, X.P sugiere utilizar una pareja de desarrolladores para que investiguen y reduzcan al máximo el riesgo que supone ese problema.

....... **Funcionalidad extra:** Nunca se debe añadir funcionalidad extra al programa aunque se piense que en un futuro será utilizada. Sólo el 10% de la misma es utilizada, lo que implica que el desarrollo de funcionalidad extra es un desperdicio de tiempo y recursos.

....... **Refactorizar.** Refactorizar es mejorar y modificar la estructura y codificación de códigos ya creados sin alterar su funcionalidad. Refactorizar supone revisar de nuevo estos códigos para procurar optimizar su funcionamiento. Es muy común rehusar códigos ya creados que contienen funcionalidades que no serán usadas y diseños obsoletos. Esto es un error porque puede generar código completamente inestable y muy mal diseñado; por este motivo, es necesario refactorizar cuando se va a utilizar código ya creado.

.......**Tarjetas C.R.C.** El uso de las tarjetas C.R.C (Class, Responsabilities and Collaboration) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica.

.......Las tarjetas C.R.C representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad

**4.3 Pruebas**

.......Como ya se dijo en la introducción, el cliente es una parte más del equipo de desarrollo; su presencia es indispensable en las distintas fases de X.P. A la hora de codificar una historia de usuario su presencia es aún más necesaria. No olvidemos que los clientes son los que crean las historias de usuario y negocian los tiempos en los que serán implementadas. Antes del desarrollo de cada historia de usuario el cliente debe especificar detalladamente lo que ésta hará y también tendrá que estar presente cuando se realicen los test que verifiquen que la historia implementada cumple la funcionalidad especificada.

.......La codificación debe hacerse atendiendo a estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión y escalabilidad.

.......Crear test que prueben el funcionamiento de los distintos códigos implementados nos ayudará a desarrollar dicho código. Crear estos test antes nos ayuda a saber qué es exactamente lo que tiene que hacer el código a implementar y sabremos que una vez implementado pasará dichos test sin problemas ya que dicho código ha sido diseñado para ese fin. Se puede dividir la funcionalidad que debe cumplir una tarea a programar en pequeñas unidades, de esta forma se crearán primero los test para cada unidad y a continuación se desarrollará dicha unidad, así poco a poco conseguiremos un desarrollo que cumpla todos los requisitos especificados.

.......Como ya se comentó anteriormente, X.P opta por la programación en pareja ya que permite un código más eficiente y con una gran calidad.

.......X.P sugiere un modelo de trabajo usando repositorios de código donde las parejas de programadores publican cada pocas horas sus códigos implementados y corregidos junto a los test que deben pasar. De esta forma el resto de programadores que necesiten códigos ajenos trabajarán siempre con las últimas versiones. Para mantener un código consistente, publicar un código en un repositorio es una acción exclusiva para cada pareja de programadores.

.......X.P también propone un modelo de desarrollo colectivo en el que todos los programadores están implicados en todas las tareas; cualquiera puede modificar o ampliar una clase o método de otro programador si es necesario y subirla al repositorio de código. El permitir al resto de los programadores modificar códigos que no son suyos no supone ningún riesgo ya que para que un código pueda ser publicado en el repositorio tiene que pasar los test de funcionamiento definidos para el mismo.

.......La optimización del código siempre se debe dejar para el final. Hay que hacer que funcione y que sea correcto, más tarde se puede optimizar.

.......X.P afirma que la mayoría de los proyectos que necesiten más tiempo extra que el planificado para ser finalizados no podrán ser terminados a tiempo se haga lo que se haga, aunque se añadan más desarrolladores y se incrementen los recursos. La solución que plantea X.P es realizar un nuevo "Release plan" para concretar los nuevos tiempos de publicación y de velocidad del proyecto.

.......A la hora de codificar no seguimos la regla de X.P que aconseja crear test de funcionamiento con entornos de desarrollo antes de programar. Nuestros test los obtendremos de la especificación de requisitos ya que en ella se especifican las pruebas que deben pasar las distintas funcionalidades del programa, procurando codificar pensando en las pruebas que debe pasar cada funcionalidad.

**4.4 Testeo**

.......Uno de los pilares de la metodología X.P es el uso de test para comprobar el funcionamiento de los códigos que vayamos implementando.

.......El uso de los test en X.P es el siguiente:

.......Se deben crear las aplicaciones que realizarán los test con un entorno de desarrollo específico para test.

.......Hay que someter a tests las distintas clases del sistema omitiendo los métodos más triviales.

.......Se deben crear los test que pasarán los códigos antes de implementarlos; en el apartado anterior se explicó la importancia de crear antes los test que el código.

.......Un punto importante es crear test que no tengan ninguna dependencia del código que en un futuro evaluará. Hay que crear los test abstrayéndose del futuro código, de esta forma aseguraremos la independencia del test respecto al código que evalúa.

.......Como se comentó anteriormente los distintos test se deben subir al repositorio de código acompañados del código que verifican. Ningún código puede ser publicado en el repositorio sin que haya pasado su test de funcionamiento, de esta forma, aseguramos el uso colectivo del código (explicado en el apartado anterior).

.......El uso de los test es adecuado para observar la refactorización. Los test permiten verificar que un cambio en la estructura de un código no tiene porqué cambiar su funcionamiento.

.......Test de aceptación. Los test mencionados anteriormente sirven para evaluar las distintas tareas en las que ha sido dividida una historia de usuario. Para asegurar el funcionamiento final de una determinada historia de usuario se deben crear "Test de aceptación"; estos test son creados y usados por los clientes para comprobar que las distintas historias de usuario cumplen su cometido.

.......Al ser las distintas funcionalidades de nuestra aplicación no demasiado extensas, no se harán test que analicen partes de las mismas, sino que las pruebas se realizarán para las funcionalidades generales que debe cumplir el programa especificado en la descripción de requisitos

**HISTORIAS DE USUARIO, TAREAS DE INGENIERÍA Y PRUEBAS DE ACEPTACIÓN:**

|  |  |
| --- | --- |
| Historia de Usuario | |
| **Número :** 1 | **Usuario:** Administrador, usuario y empleado |
| **Nombre de la Historia**: Acceso al Sistema | |
| **Prioridad en Negocio:** Alta | **Riesgo en Desarrollo:** Media |
| **Puntos Estimado:** 7 | **Iteración Asignada: 1** |
| **Programador Responsable:** Collar Tomás – Alia Lucas | |
| **Descripción:** Los Usuarios del sistema tendrán un nombre de usuario y clave única con la que podrán ingresar al mismo Luego de haberse registrado, en lo que debe permitir la verificación de datos, por seguridad el mismo contara con solo 2 intentos posibles. En caso de sobrepasar los límites el usuario se deberá dirigir presencialmente a la veterinaria para solicitar el cambio de contraseña. | |

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| **Número de Tarea:** 1 | **Número de Historia:** #1 –Acceso al sistema |
| **Nombre de la Tarea**: Diseño de Interfaz de usuario (GUI) de Acceso al Sistema | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados:** 5 |
| **Fecha Inicio:** 10/04/2020 | **Fecha Fin: 15**/04/2020 |
| **Programador Responsable:** Collar Tomás – Alia Lucas | |
| **Descripción:** Desarrollar una Interfaz Gráfica en la que los usuarios puedan Acceder al Sistema ingresando sus correspondientes datos para el inicio de sesión.  NOTA: Para los usuarios pertenecientes al Profesional de Salud Veterinaria se les asignara automáticamente. | |

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| **Número de Tarea:** 2 | **Número de Historia:** #1 –Acceso al sistema |
| **Nombre de la Tarea**: Autenticación de Usuarios | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados:** 7 |
| **Fecha Inicio:** 15/04/2020 | **Fecha Fin:** 21/04/2020 |
| **Programador Responsable:** Collar Tomás – Alia Lucas | |
| **Descripción:** Desarrollar un sistema que permita hacer la verificación de los datos ingresados por el usuario, del cual permitirá el ingreso siempre que los datos ingresados corresponden con los de la base de datos. | |

|  |  |
| --- | --- |
| Caso de Prueba de Aceptación | |
| **Código:** 1 | **Número de Historia:** #2 – Autenticación de Usuario |
| **Nombre**: Prueba del Sistema de Ingreso de usuarios | |
| **Condiciones de Ejecución:**  El usuario debió previamente haberse registrado en el sistema  El usuario se acuerda Perfectamente de su usuario y contraseña  El usuario desea hacer uso de las funcionalidades del sistema | |
| **Entrada/Paso de Ejecución:**  1- El usuario Ingresa los Datos que el Sistema le solicita  2- El usuario solicita hacer uso de alguna funcionalidad | |
| **Resultado Esperado:**  1- El sistema recibe los datos y verifica en la Base de Datos si los datos Ingresados son correctos.  2- Los datos son correctos y se accede perfectamente  3- El sistema habilita las funciones , ya que los datos ingresados fueron correctos | |
| Evaluación de la Prueba: Satisfactorio | |

|  |  |
| --- | --- |
| Caso de Prueba de Aceptación | |
| **Código:** 2 | **Número de Historia:** #2 – Autenticación de Usuario |
| **Nombre**: Prueba del Sistema de Ingreso de usuarios | |
| **Condiciones de Ejecución:**  El usuario debió previamente haberse registrado en el sistema  El usuario **NO** se acuerda Perfectamente de su usuario y contraseña | |
| **Entrada/Paso de Ejecución:**  1- El usuario Ingresa los Datos que el Sistema le solicita  2- El usuario insiste en volver a ingresar sus datos nuevamente | |
| **Resultado Esperado:**  1- El sistema recibe los datos y verifica en la Base de Datos si los datos Ingresados son correctos.  2- Los datos son erróneos y se rechaza el ingreso  3- El sistema bloquea el ingreso del mismo y imprime un mensaje en pantalla.  Ej.: “Por cuestiones de seguridad se ha bloqueado el acceso, acérquese presencialmente a la veterinaria para reactivar su cuenta” | |
| Evaluación de la Prueba: Satisfactorio | |

|  |  |
| --- | --- |
| Historia de Usuario | |
| **Número :** 2 | **Usuario:** Administrador, usuario y empleado |
| **Nombre de la Historia**: Registro de turnos. | |
| **Prioridad en Negocio:** Alta | **Riesgo en Desarrollo:** Media |
| **Puntos Estimado:** 7 | **Iteración Asignada: 1** |
| **Programador Responsable:** Collar Tomás – Alia Lucas | |
| **Descripción:** Como usuario del Sistema solicitó registrar turno cuando tenga la necesidad de que atiendan a mi mascota siempre cuando lo desee, con una fecha y una hora asignada que él mismo me ofrezca. | |

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| **Número de Tarea:** 3 | **Número de Historia:** #2 – Registro de Turnos |
| **Nombre de la Tarea**: Reservar el turno del cliente | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados:** 7 |
| **Fecha Inicio:** 21/04/2020 | **Fecha Fin:** 26/04/2020 |
| **Programador Responsable:** Collar Tomás – Alia Lucas | |
| **Descripción:** Desarrollar un sistema que permita reservar un turno cuando el cliente lo solicite, ofreciéndole el día y la hora que estén disponibles a su disposición. | |

|  |  |
| --- | --- |
| Caso de Prueba de Aceptación | |
| **Código:** 3 | **Número de Historia:** #2 – Registro de Turnos |
| **Nombre**: Prueba del módulo Reserva de Turnos | |
| **Condiciones de Ejecución:**  El usuario debió autenticarse exitosamente en el sistema antes de hacer uso del dicho módulo  El usuario tiene la necesidad de llevar a su mascota al veterinario | |
| **Entrada/Paso de Ejecución:**  1- El usuario solicita la opción registrar turno  2- El usuario ingresa la fecha y hora disponibles que el sistema le ofrece | |
| **Resultado Esperado:**  1- El Sistema recibe la petición y carga automáticamente todos los turnos que NO estén reservados por otros clientes y muestra en pantalla  2- El Sistema reserva el turno, luego actualiza el turno del cliente a ocupado y lo guarda en la base de datos. | |
| Evaluación de la Prueba: Satisfactorio | |

|  |  |
| --- | --- |
| Historia de Usuario | |
| **Número :** 3 | **Usuario:** Administrador, usuario y empleado |
| **Nombre de la Historia**: Cancelación de Turnos | |
| **Prioridad en Negocio:** Alta | **Riesgo en Desarrollo:** Alta |
| **Puntos Estimado:** 2 | **Iteración Asignada: 1** |
| **Programador Responsable:** Collar Tomás – Alia Lucas | |
| **Descripción:** Como usuario del Sistema solicitó la funcionalidad de poder cancelar el turno en el caso de que por motivos personales no pueda asistir al consultorio veterinario | |

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| **Número de Tarea:** 4 | **Número de Historia:** #3 –Cancelación de Turnos |
| **Nombre de la Tarea**: Reservar el turno del cliente | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados:** 7 |
| **Fecha Inicio:** 26/04/2020 | **Fecha Fin:** 28/04/2020 |
| **Programador Responsable:** Collar Tomás – Alia Lucas | |
| **Descripción:** Desarrollar en el sistema un módulo que permita al cliente poder cancelar un turno siempre y cuando no se excedan las 24 horas hábiles una vez realizado el turno. | |

|  |  |
| --- | --- |
| Caso de Prueba de Aceptación | |
| **Código:** 4 | **Número de Historia:** #3 – Cancelación de Turnos |
| **Nombre**: Prueba del módulo Suprimir Turnos | |
| **Condiciones de Ejecución:**  El usuario debe autenticarse exitosamente en el sistema antes de hacer uso del dicho módulo  El usuario debió previamente haber reservado un turno  El usuario no puede concurrir el día que solicitó el turno por contingencias ocurridas | |
| **Entrada/Paso de Ejecución:**  1- El usuario entra a la funcionalidad Mis Turnos  2- El usuario solicita la opción cancelar turno | |
| **Resultado Esperado:**  1- El Sistema recibe la petición y compara si todavía no transcurrieron las 24 horas previstas  2- El Sistema da de baja el turno el turno exitosamente | |
| Evaluación de la Prueba: Satisfactorio | |

|  |  |
| --- | --- |
| Caso de Prueba de Aceptación | |
| **Código:** 5 | **Número de Historia:** #3 – Cancelación de Turnos |
| **Nombre**: Prueba del módulo Suprimir Turnos | |
| **Condiciones de Ejecución:**  El usuario debió autenticarse exitosamente en el sistema antes de hacer uso del dicho módulo  El usuario debió previamente haber reservado un turno  El usuario no puede concurrir el día que solicitó el turno por contingencias ocurridas | |
| **Entrada/Paso de Ejecución:**  1- El usuario entra a la funcionalidad Mis Turnos  2- El usuario solicita la opción cancelar turno | |
| **Resultado Esperado:**  1- El Sistema recibe la petición y compara si todavía no transcurrieron las 24 horas previstas  2- El Sistema anula la opción, ya que transcurrieron más de 24 horas y puede afectar a la planificación | |
| Evaluación de la Prueba: Satisfactorio | |

|  |  |
| --- | --- |
| Historia de Usuario | |
| **Número :** 4 | **Usuario:** Solo Administrador |
| **Nombre de la Historia**: Gestión de Pacientes | |
| **Prioridad en Negocio:** Alta | **Riesgo en Desarrollo:** Alta |
| **Puntos Estimado:** 5 | **Iteración Asignada: 1** |
| **Programador Responsable:** Collar Tomás – Alia Lucas | |
| **Descripción:** Como usuario del Sistema solicitó la funcionalidad de poder buscar un cliente, modificar y también borrar | |

|  |  |
| --- | --- |
| Tarea de Ingeniería | |
| **Número de Tarea:** 5 | **Número de Historia:** #4 – Gestión de Pacientes |
| **Nombre de la Tarea**: Añadir funcionalidades al Administrador | |
| **Tipo de Tarea:** Desarrollo | **Puntos Estimados:** 7 |
| **Fecha Inicio:** 26/04/2020 | **Fecha Fin:** 28/04/2020 |
| **Programador Responsable:** Collar Tomás – Alia Lucas | |
| **Descripción:** Desarrollar en el sistema que permita al administrador poder gestionar los pacientes ofreciendo las opciones de consultar, borrar y añadir | |

|  |  |
| --- | --- |
| Caso de Prueba de Aceptación | |
| **Código:** 6 | **Número de Historia:** #4 – Gestión de Pacientes |
| **Nombre**: Prueba del módulo para administrador | |
| **Condiciones de Ejecución:**  El usuario debió autenticarse exitosamente en el sistema antes de hacer uso del dicho módulo  El usuario debe ser un tipo de usuario especial | |
| **Entrada/Paso de Ejecución:**  1- El usuario entra a la funcionalidad Gestión de Pacientes  2- El usuario desea ejecutar alguna de las opciones que le ofrece la funcionalidad | |
| **Resultado Esperado:**  1- El Sistema recibe la petición y verifica si el usuario es un tipo de usuario especial  2- El sistema habilita el módulo luego de que la verificación sea correcta  3- El usuario ejecuta exitosamente la opción que le solicitó ejecutar | |
| Evaluación de la Prueba: Satisfactorio | |

|  |  |
| --- | --- |
| Caso de Prueba de Aceptación | |
| **Código:** 7 | **Número de Historia:** #4 – Gestión de Pacientes |
| **Nombre**: Prueba del módulo para administrador | |
| **Condiciones de Ejecución:**  El usuario debió autenticarse exitosamente en el sistema antes de hacer uso del dicho módulo  El usuario debe ser un tipo de usuario especial | |
| **Entrada/Paso de Ejecución:**  1- El usuario entra a la funcionalidad Gestión de Pacientes  2- El usuario desea ejecutar alguna de las opciones que le ofrece la funcionalidad | |
| **Resultado Esperado:**  1- El Sistema recibe la petición y verifica si el usuario es un tipo de usuario especial  2- El sistema anula el módulo luego de que la verificación sea incorrecta  3- El sistema no permitirá al usuario hacer uso del dicho módulo | |
| Evaluación de la Prueba: Satisfactorio | |

|  |  |
| --- | --- |
| **HISTORIA DE USUARIO** | |
| **Número:** 4 | **Usuario: Administrador** |
| **Nombre Historia:** Gestión de Historias Clínicas | |
| **Prioridad en Negocio:** Alta | **Riesgo en Desarrollo:** media |
| **Puntos  Estimados: 2** | **Iteración Asignada: 1** |
| **Programador Responsable:** Alia Lucas. | |
| **Descripción:**  El sistema permitirá al usuario, modificar, visualizar, eliminar o crear una historia clínica. | |

**5 Análisis de Riesgos**

El análisis de riesgos durante el desarrollo del proyecto es un proceso que comprende la identificación de activos informáticos, sus vulnerabilidades y amenazas a los que se encuentran expuestos así como su probabilidad de ocurrencia y el impacto de las mismas, a fin de determinar los controles adecuados para poder prevenirlos antes de que suceda.

**Riesgo de Producto**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Escenario | **Descripción** | **Tipo** | **Efecto** | **Como prever** |
| #1 | Bajo rendimiento de la herramienta CASE, no tiene las funciones esperadas o no están disponibles todas | media | serio | Se abonará el pago de la herramienta para proceder a la versión premium y hacer uso completo de todas las funciones |
| #2 | El código generado de la herramienta CASE no es eficiente | baja | tolerable | Reorganizar el equipo de Desarrolladores para que uno se encargue de hacerlo manualmente |

**Riesgo de Proyecto**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Escenario | **Descripción** | **Tipo** | **Efecto** | **Como prever** |
| #1 | Renuncia de algún programador experimentado y líder del equipo antes de finalizar el proyecto | alta | seria | Se motivara al programador con un salario mayor al de los demás por su antigüedad y experiencia |
| #2 | Falla de las Herramientas para realizar el Proyecto, como pueden ser los equipos Informáticos | Baja | Seria | Realizar Mantenimiento preventivo de los Equipos Informáticos una vez por semana |

**Riesgo de Negocio**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Escenario | **Descripción** | **Tipo** | **Efecto** | **Como prever** |
| #1 | El cliente no está satisfecho con el Software o no era lo que esperaba | alta | catastrófica | Se deberá hacer énfasis y mayor prioridad en la recogida de información y consultar los documentos de la organización para determinar qué es lo que el cliente realmente quiere o necesita |

**6 Entrevista**

Para la recogida de información utilizamos Entrevista en Estructura Embudo

**6.1 Preguntas Abiertas**

¿Cual es la Modalidad en la que atienden a los pacientes?

¿Cómo podría describir las creencias de este establecimiento?

¿Que tan próximas son las reuniones con el paciente si el mismo está realizando un tratamiento y necesita venir muy seguido?

¿Qué es lo que espera a futuro de su establecimiento luego de implementar este sistema de información?

¿Cual es la Patología común que encuentra en la mayoría de sus pacientes?

**6.2 Preguntas Cerradas**

En caso de que se atienden por orden de llegada ¿Dan prioridad a pacientes de riesgo?

¿Cuántos Profesionales de Salud Veterinaria se encuentran trabajando actualmente en este establecimiento?

¿En qué tipo de Animales tiene más demanda?

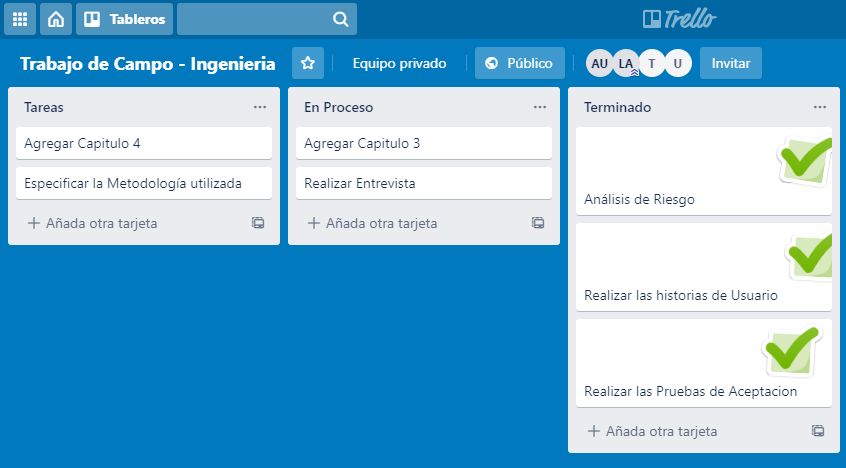
¿Cuantos sponsors tiene?

**6.3 Preguntas de sondeo**

Para que un paciente entre en internación ¿Cuales son los síntomas mínimos que debe tener el mismo?

**7 Herramientas Utilizadas:**

* Googledocs
* Trello



**8 Bibliografía:**

<https://trello.com/b/eLtjuu2C/trabajo-de-campo-ingenieria>

<http://programacionextrema.tripod.com/fases.htm>

<http://oness.sourceforge.net/proyecto/html/ch05s02.html>

<https://www.goconqr.com/es/mindmap/3207836/fases-metodolog-a-xp-programaci-n-extrema->

METODOLOGIA ÁGIL DE DESARROLLO DE SOFTWARE PROGRAMACION

EXTREMA (pdf de aplicación práctica de la Universidad Nacional Autónoma de Nicaragua).