GUIA 1 TALLER DE PROGRAMACION

JUAN DAVID SERRANO VACCA

UNIVERSIDAD MANUELA BELTRAN

INGENIERIA DE SOFTWARE

08/2024

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Identificador: RF01 Nombre: Gestión de bovino** | | | | **Requerimiento que lo Utiliza o Especializa:**  R1.1 | **Documentos de visualización Asociados:** | | | **Entrada:**  Raza: String (20)  Edad: Integer (0-30 años)  Sexo: String (1) ('M' o 'F')  Historial Médico: String (variante)  Registros de vacunación: String (variante)  Identificación individual: String (15) | | **Salida:**  Verificación de la creación de la ficha del bovino en la base de datos  Variables de salida: Confirmación de la ficha creada con los datos capturados (raza, edad, sexo, etc.) | | **Descripción:**  El software debe permitir almacenar una ficha única para cada bovino, incluyendo información relevante como raza, edad, sexo, historial médico, entre otros. Se deben realizar validaciones básicas en la entrada de datos (por ejemplo, edad debe ser un número entero entre 0 y 30). | | | | **Manejo de Situaciones Anormales**  Violaciones a la precondición:  -Entrada de datos no válida (por ejemplo, raza vacía o edad fuera de rango). El sistema debe rechazar la entrada y notificar al usuario.  Situaciones anormales con respecto a la utilización de recursos:  -Si no hay suficiente espacio en disco para guardar la ficha, el sistema debe notificar al usuario y no permitir la creación hasta que se resuelva la situación. | | | | **Criterios de Aceptación**   1. Para las entradas válidas, la ficha del bovino debe ser creada y almacenada correctamente en la base de datos. 2. Para las situaciones anormales, el sistema debe comportarse como se especifica, manejando correctamente las entradas inválidas y problemas de recursos. 3. Se debe verificar que la información guardada es persistente y accesible de manera segura, cumpliendo con las restricciones de privacidad y seguridad de datos. 4. Se debe garantizar que la identificación individual del bovino sea única y no se pueda duplicar. | | | |

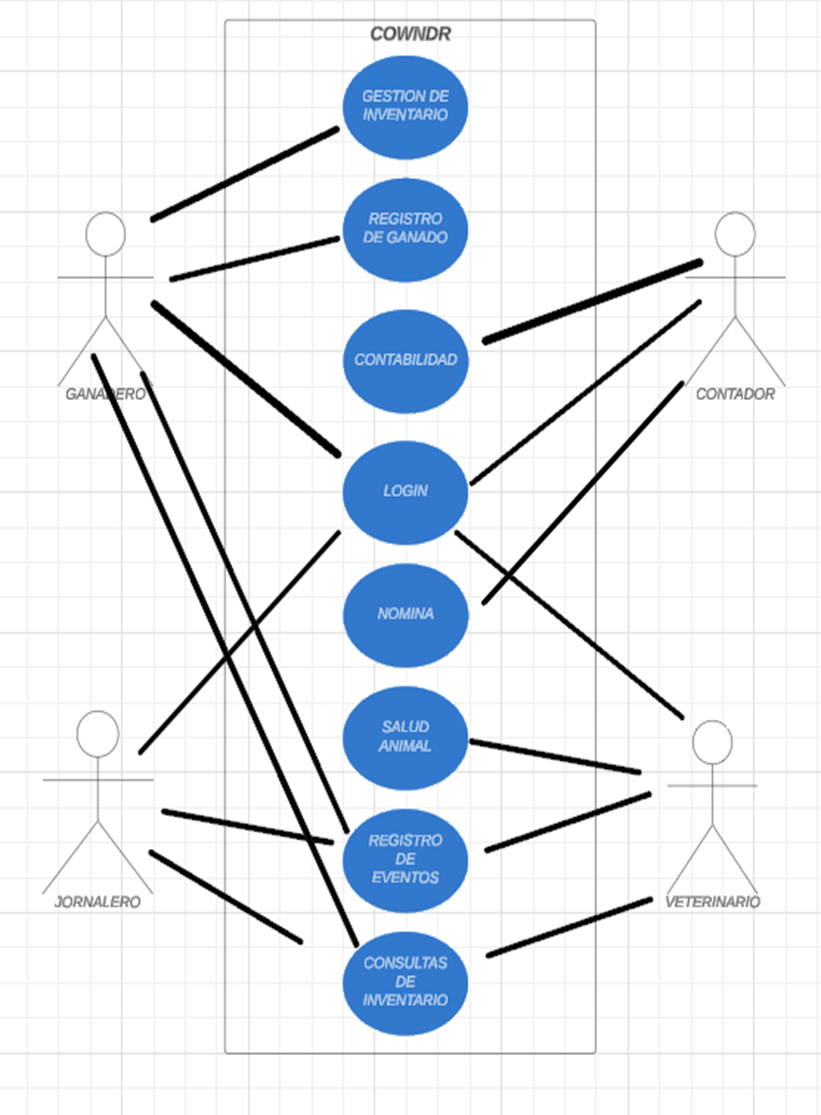
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Identificador: R2 Nombre: Gestión de potreros** | | | | **Requerimiento que lo Utiliza o Especializa:**  R2.1 | **Documentos de visualización Asociados:** | | | **Entrada:**  Dimensiones del potrero: Float (0.1 - 1000) (en metros cuadrados)  Límites del potrero: Coordenadas geográficas o un identificador de zona  Nombre del potrero: String (50)  Capacidad máxima de animales: Integer (1-1000) | | **Salida:**  Verificación de la creación del potrero en el sistema | | **Descripción:**  El software debe permitir crear potreros virtuales para organizar y gestionar la distribución de los animales. El potrero debe incluir información como dimensiones, límites, un nombre identificador y la capacidad máxima de animales. Se deben realizar validaciones en las entradas para asegurar que las dimensiones sean realistas y que la capacidad máxima no exceda los límites permitidos | | | | **Manejo de Situaciones Anormales**  Violaciones a la precondición:  Dimensiones fuera de rango o límites del potrero no válidos. El sistema debe rechazar la entrada y notificar al usuario. | | | | **Criterios de Aceptación**   1. Para las entradas válidas, el potrero debe ser creado y almacenado correctamente en el sistema. 2. Para las situaciones anormales, el sistema debe comportarse como se especifica, manejando correctamente las entradas inválidas y problemas de recursos. 3. Se debe verificar que la información guardada es persistente y que la capacidad máxima de animales por potrero se respeta durante la asignación de animales. 4. Se debe garantizar que el software pueda manejar múltiples potreros sin degradar el rendimiento ni afectar otras funcionalidades. | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Identificador: R2.2 Nombre: Traslado de animales** | | | | **Requerimiento que lo Utiliza o Especializa:**  R2 | **Documentos de visualización Asociados:** | | | **Entrada:**  Identificador del animal: String (15)  Potrero de origen: String (50)  Potrero de destino: String (50)  Fecha del movimiento: Date (YYYY-MM-DD)  Motivo del movimiento: String (200) | | **Salida:**  Actualización del registro de ubicación del animal en el sistema  Variables de salida: Confirmación del traslado del animal con detalles de fecha, potrero de origen y destino, y motivo | | **Descripción:**  El software debe permitir trasladar animales de un potrero a otro, así como agregar y eliminar animales del inventario, manteniendo un registro actualizado de su ubicación. Se deben capturar y almacenar detalles como la fecha y motivo del movimiento, y generar notificaciones de cambios en la ubicación de los animales. Este requisito se especializa en la "Gestión de potreros" (R1), ya que depende de la correcta administración de estos espacios. | | | | **Manejo de Situaciones Anormales**  Violaciones a la precondición:  Potrero de destino no válido o inexistente. El sistema debe rechazar la operación y notificar al usuario.  Situaciones anormales con respecto a la utilización de recursos:  Si no se puede actualizar la ubicación del animal en el sistema debido a problemas técnicos, el sistema debe notificar al usuario y asegurar que los cambios no se guarden hasta que el problema sea resuelto. | | | | **Criterios de Aceptación**   1. Para las entradas válidas, el traslado o eliminación del animal debe ser registrado correctamente en el sistema. 2. Para las situaciones anormales, el sistema debe comportarse como se especifica, manejando correctamente las entradas inválidas y problemas de recursos. 3. Se debe verificar que la información sobre la ubicación del animal es persistente y se actualiza correctamente en tiempo real. 4. Se debe garantizar que las restricciones de bioseguridad y salud animal se respeten en cada movimiento, notificando cualquier incumplimiento al usuario | | | |

|  |  |  |
| --- | --- | --- |
| **Identificador: R4 Nombre: Modulo Contabilidad** | | |
| **Requerimiento que lo Utiliza o Especializa:**  R4 | **Documentos de visualización Asociados:** | |
| **Entrada:**  salarios: Decimal (con dos decimales)  deducciones: Decimal (con dos decimales) | | **Salida:**  Recibo de pago generado |
| **Descripción:**  El módulo de contabilidad debe permitir el registro de salarios y horas trabajadas de los empleados, generar recibos de pago automáticamente, y calcular impuestos y deducciones de manera precisa. La función debe cumplir con las regulaciones fiscales y laborales locales y asegurar la protección de datos financieros y de empleados. | | |
| **Manejo de Situaciones Anormales**  Violaciones a la precondición:  Si los datos de entrada están incompletos o son incorrectos (por ejemplo, salarios negativos), el sistema debe mostrar un mensaje de error y no procesar la información.  Situaciones anormales con respecto a la utilización de recursos:  Si hay falta de espacio en disco, el sistema debe mostrar un mensaje de error y no guardar los datos. | | |
| **Criterios de Aceptación**   1. Para las entradas válidas:   La salida debe ser un recibo de pago preciso y un reporte detallado que incluya todos los cálculos correctos.   1. Para situaciones anormales:   El comportamiento debe ser el especificado en el ítem de manejo de situaciones anormales.   1. Verificación persistente:   Los datos de salarios y pagos deben estar correctamente almacenados en la base de datos y ser accesibles para su revisión.   1. Restricciones de desempeño:   El sistema debe generar recibos y reportes en menos de 5 segundos bajo condiciones normales de uso.  El sistema debe manejar un máximo de 500 registros por minuto sin degradación del desempeño. | | |

|  |  |  |
| --- | --- | --- |
| **Identificador: R5 Nombre: Autenticacion de Usuario** | | |
| **Requerimiento que lo Utiliza o Especializa:**  R5.1 | **Documentos de visualización Asociados:** | |
| **Entrada:**  usuario: String (50 caracteres)  contraseña: String (50 caracteres) | | **Salida:**  Confirmación de acceso exitoso o mensaje de error en caso de credenciales inválidas |
| **Descripción:**  El sistema debe permitir a los usuarios autenticarse mediante un formulario de inicio de sesión. El usuario debe ingresar un nombre de usuario y una contraseña. El sistema debe validar estas credenciales comparándolas con las almacenadas en la base de datos. Si las credenciales son correctas, el sistema permitirá el acceso al usuario; de lo contrario, mostrará un mensaje de error. | | |
| **Manejo de Situaciones Anormales**  Violaciones a la precondición:  Si el nombre de usuario o la contraseña están vacíos, el sistema debe mostrar un mensaje de error indicando que los campos no pueden estar vacíos.  Si el nombre de usuario o la contraseña no coinciden con los almacenados en la base de datos, el sistema debe mostrar un mensaje de error indicando credenciales incorrectas.  Situaciones anormales con respecto a la utilización de recursos:  Si hay problemas con la base de datos (por ejemplo, está caída), el sistema debe mostrar un mensaje de error indicando que no se puede verificar la autenticidad en este momento. | | |
| **Criterios de Aceptación**   1. Para las entradas válidas:   El sistema debe permitir el acceso al usuario si las credenciales son correctas y redirigir a la página principal del sistema.   1. Para situaciones anormales:   El sistema debe mostrar mensajes de error apropiados para las credenciales inválidas o problemas de conexión, como se especifica en el manejo de situaciones anormales.   1. Verificación persistente:   El sistema debe registrar intentos de inicio de sesión, exitosos y fallidos, para auditoría y análisis.   1. Restricciones de desempeño:   El proceso de autenticación debe completarse en menos de 3 segundos bajo condiciones normales de uso. | | |

2.UML



|  |
| --- |
| **Preguntas Orientadoras** |
| En este apartado se realiza el análisis de los datos obtenidos, estos pueden ser de forma cualitativa o cuantitativa según la naturaleza de la práctica.  ¿Cuáles fueron los aprendizajes obtenidos al realizar esta guía?, liste como mínimo 3 aprendizajes y relaciónelos con su futuro que hacer profesional.   1. Se aprendió nuevos requisitos para la documentación de requisitos funcionales. 2. Se siguen viendo diagramas UML que son esenciales para los ingenieros de software. 3. Creatividad para idear soluciones a problemas de la vida real para así desarrollar un sistema de información.   ¿Dónde presento mayor dificultad resolviendo la guía? y ¿cómo lo resolvieron? ¿cuáles fueron las estrategias de solución?   1. Tratando de entender que tocaba hacer en la guía ya que para mí la guía es un tanto compleja de entender y saber cuales son los entregables y la forma de entregarlos. |

**Actividad de trabajo independiente**

Consulta bibliográfica en bases de datos digitales.

Realiza la búsqueda de información sobre novedades en programación.

**1. Lenguajes de Programación para la Inteligencia Artificial**

**Novedad**: **C++ con soporte para AI y Machine Learning**

* **Descripción**: C++ ha evolucionado para ofrecer mejor soporte en el desarrollo de aplicaciones de inteligencia artificial y machine learning. La introducción de bibliotecas y frameworks específicos como Eigen para álgebra lineal y Dlib para machine learning ha facilitado la implementación de modelos complejos en C++. Además, se está trabajando en nuevas extensiones y herramientas que integran capacidades de AI directamente en el lenguaje.
* **Impacto**: La capacidad de C++ para trabajar de manera más eficiente con datos y modelos de machine learning lo hace atractivo para desarrolladores que buscan optimización en el rendimiento y velocidad en aplicaciones de AI. Esto es particularmente relevante en aplicaciones que requieren un procesamiento intensivo, como los videojuegos o simulaciones científicas.

**2. Avances en Programación Asistida por IA**

**Novedad**: **Integración de Inteligencia Artificial en IDEs**

* **Descripción**: Los entornos de desarrollo integrado (IDEs) están incorporando herramientas impulsadas por IA para asistir en la codificación. Por ejemplo, herramientas como GitHub Copilot, desarrollada por OpenAI y GitHub, utilizan modelos de lenguaje para sugerir líneas de código y completar fragmentos de código automáticamente mientras el desarrollador escribe. Esto ayuda a mejorar la productividad y reducir errores en la codificación.
* **Impacto**: La programación asistida por IA está revolucionando cómo los desarrolladores interactúan con sus entornos de trabajo, haciendo que el proceso de escritura de código sea más eficiente y menos propenso a errores. Esto también puede reducir la curva de aprendizaje para los nuevos desarrolladores al proporcionar sugerencias y ejemplos contextuales en tiempo real.

Realice la lectura del siguiente artículo: Keepcoding Tech School. (2024). ¿Qué son los paradigmas de programación y qué tipos hay? https://keepcoding.io/blog/paradigmas-de-programacion/ (Recurso de Internet)

Enuncie y explique brevemente los paradigmas de programación propuestos en dicho artículo, investigando la información solicitada en el siguiente cuadro comparativo.

|  |  |  |  |
| --- | --- | --- | --- |
| **Paradigma de Programación** | **Ventajas** | **Desventajas** | **Lenguaje de programación** |
| **Imperativo** | - Control detallado del flujo de ejecución. - Facilita la optimización de código y manejo eficiente de recursos. | - Complejidad y dificultad de mantenimiento con el crecimiento del código. - Errores difíciles de rastrear por el manejo del estado mutable. | C, C++, Python, Java |
| **Declarativo** | - Simplifica el código al enfocarse en "qué hacer" en lugar de "cómo hacerlo". - Menos riesgo de errores relacionados con el estado. | - Menor control sobre los detalles de la ejecución. - Posible rendimiento inferior en comparación con paradigmas imperativos. | SQL, HTML, Prolog |
| **Orientado a Objetos (POO)** | - Promueve la modularidad y reutilización de código. - Facilita la modificación y extensión del código. | - Puede introducir sobrecarga adicional en memoria y procesamiento. - Curva de aprendizaje más pronunciada. | Java, C++, Python, Ruby |
| **Funcional** | - Inmutabilidad de datos lleva a menos errores y código más predecible. - Facilita el razonamiento y prueba del código. | - Difícil para quienes están acostumbrados a paradigmas imperativos. - Posible impacto en el rendimiento debido a la inmutabilidad y recursión. | Haskell, Lisp, Scala |
| **Lógico** | - Permite expresar conocimientos y resolver problemas mediante reglas lógicas. - Inferencia automática de soluciones. | - Menos eficiente para ciertos problemas. - Complejidad que puede dificultar la comprensión y mantenimiento del código. | Prolog |
|  |  |  |  |

LINK VIDEO

https://youtu.be/RH0Cl0KHXhk

BIBLIOGRAFIA

<https://keepcoding.io/blog/paradigmas-de-programacion/>

<https://uxcale.com/integracion-de-ia-en-desarrollo-de-software/>

https://learn.microsoft.com/es-es/windows/ai/windows-ml/get-started-desktop