PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
FACULTAD DE INGENIERÍA
ESCUELA DE SISTEMAS

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERIA DE SISTEMAS Y COMPUTACIÓN

"DESARROLLO DE UN SISTEMA DE MONITOREO EN EL DEPARTAMENTO DE CONSERVACIÓN DE PALMA AFRICANA EN EL INSTITUTO NACIONAL AUTÓNOMO DE INVESTIGACIONES AGROPECUARIAS SEDE SANTO DOMINGO APLICANDO LA METODOLOGÍA EXTREME PROGRAMMING UTILIZANDO LA TÉCNICA DE PAIR PROGRAMMING"

RICARDO ARMAS
DANIELA LARREA

DIRECTOR: ING. FABIÁN DE LA CRUZ

QUITO, OCTUBRE 2012

Queremos brindar un agradecimiento especial a nuestros padres que nos han apoyado siempre desde que comenzamos esta etapa de nuestras vidas que al fin la estamos cerrando. Ellos han estado presentes y pendientes de todo y estamos seguros de que seguirán haciéndolo. Por eso, mil y una vez gracias, a ustedes, la razón por la que hoy estamos cumpliendo con este ciclo, no podríamos haberlo hecho sin su apoyo incondicional.

Igualmente, queremos agradecer a todas las personas que formaron parte de este proyecto, y que gracias a que nos brindaron su ayuda y orientación, pudimos concluir con el mismo.

Los Autores

TABLA DE CONTENIDOS

TABLA DE CONTENIDOS ____ *2*

1. ASPECTOS PRELIMINARES ____ *4*

1.1. Resumen ____ *4*

1.2. Introducción ____ *4*

2. MARCO TEÓRICO ____ *8*

2.1. Metodologías ____ *8*

2.1.1. Metodologías Orientadas a Objetos ____ *8*

2.1.2. Metodologías Ágiles ____ *10*

2.1.3. Extreme Programming ____ *13*

2.2. Herramientas ____ *32*

2.2.1. Lenguajes de Modelamiento UML ____ *33*

2.3. Lenguajes de Programación ____ *39*

2.3.1. Programación Orientada a Objetos ____ *41*

2.3.2. Lenguaje de Programación JAVA ____ *45*

2.4. Bases de Datos ____ *48*

2.4.1. Tipos de Bases de Datos ____ *49*

2.4.2. Modelos de Bases de Datos ____ *50*

2.4.3. Sistema Gestor de Base de Datos ____ *52*

2.4.4. Administrador de la Base de Datos ____ *53*

2.4.5. Microsoft SQL-Server ____ *53*

2.5. Selección para uso en Caso Práctico ____ *56*

3. CASO DE ESTUDIO ____ *62*

3.1 Presentación de la Empresa ____ *62*

3.2 Recolección de Datos ____ *64*

3.2.1 Justificación ____ *64*

3.2 Antecedentes _____ *65*

3.2.1 Proceso de Producción de semilla de Palma Africana _____ *66*

3.2.2 Equipos y Parámetros en los cuartos _____ *72*

3.2.3 Sistema Electrónico SCADA _____ *76*

4. DESARROLLO DEL PRODUCTO _____ *86*

4.1 Diagramas Casos de Uso _____ *86*

4.2 Diagramas de Secuencias _____ *87*

4.3 Diagramas de Actividades _____ *99*

4.4 Historias de Usuario _____ *100*

5. CONCLUSIONES Y RECOMENDACIONES _____ *112*

6. BIBLIOGRAFÍA _____ *115*

1. ASPECTOS PRELIMINARES

1.1. Resumen

En la actual disertación, se desarrolló un sistema de monitoreo para el departamento de conservación de palma africana del INIAP, aplicando la metodología de Extreme Programming y utilizando la técnica de Pair Programming. El sistema aporta en la toma de decisiones finales, ya que permite analizar los valores obtenidos de temperatura y humedad en las diferentes etapas de la investigación mediante la generación de reportes con datos actuales e históricos. Este se comunica con un sistema electrónico SCADA, el cual registra en intervalos de tiempo determinados, datos sobre las condiciones del lugar en cuestión. Por lo tanto esta base interactúa con los dos sistemas.

La interfaz gráfica que se presenta al usuario final es amigable y cómoda. Permite elegir intervalos de tiempos para la generación de los reportes de monitoreo, así como también los cuartos específicos en los que se generaron los datos. Igualmente permite dar un nivel para las alertas que genera el sistema de acuerdo a los parámetros ingresados.

El sistema aporta un valor agregado a la institución por el hecho de generar estos datos estadísticos que van directamente relacionados con la toma de decisiones sobre la manera en que se está llevando a cabo uno u otro proceso y también sobre quien o quienes son los encargados de los mismos.

1.2. Introducción

En el presente proyecto se desarrolla el sistema de monitoreo para el departamento de conservación de palma africana en el Instituto Nacional Autónomo de Investigaciones Agropecuarias Sede Santo Domingo, con la aplicación de la metodología de Extreme Programming y utilizando la técnica de Pair Programming, para realizar el seguimiento de los valores obtenidos de

temperatura y humedad en las diferentes etapas de la investigación mediante la generación de reportes con datos actuales e históricos, necesarios para la toma de decisiones.

Para el presente trabajo se ha divido en los siguientes capítulos:

- Capítulo 1: Introducción al proyecto, agradecimientos, dedicatoria, resumen ejecutivo.
- Capítulo 2: Donde se explican las metodologías y herramientas necesarias para desarrollar el proyecto. En este capitulo, se empieza dando una introducción sobre las metodologías orientadas a objetos seguido por una explicación de como funciona la metodología de extreme programming y la técnica de pair programming, los conceptos y reglas a seguir.

Se explica también sobre las herramientas utilizadas en el desarrollo, iniciando con una descripción de los lenguajes de modelamiento UML, seguido por los lenguajes de programación de manera general pero profundizando en la programación orientada a objetos y en el lenguaje utilizado JAVA.

Se habla sobre bases de datos, tipos, modelos, sistema gestor de base de datos y se pone énfasis en la explicación en Microsoft SQL Server que se utilizó en el desarrollo del presente producto.

Por último se realiza un análisis sobre la razón y beneficios de la selección de la metodología, técnica, herramientas y base de datos.

- Capítulo 3: Este capítulo esta destinado a la descripción del caso de estudio, se hace la presentación de la empresa explicando sus actividades y características más importantes, se presenta la justificación para el desarrollo realizado y los antecedentes que llevaron al mismo.

Posteriormente se realiza una explicación sobre el Proceso de Producción de semilla de Palma Africana que se realiza en la sede del INIAP en Santo

Domingo, como se lo realiza, en que condiciones, los equipos utilizados en el proceso y las condiciones de fallo que pueden ocurrir en determinadas situaciones.

Se habla también sobre el sistema electrónico utilizado actualmente para controlar los equipos y registrar la información obtenida en la base de datos, como es su funcionamiento y sus características.

- Capítulo 4: Se muestran los diagramas realizados: casos de uso, actividades y secuencia para los diferentes módulos y procesos del sistema.

CAPITULO II

MARCO TEÓRICO

2. MARCO TEÓRICO

El siguiente capítulo explica las metodologías y herramientas necesarias para desarrollar un sistema de software. La teoría que a continuación se va a revisar será fundamental para la selección de la metodología y técnica a aplicar, así como también de las herramientas que se utilizarán a lo largo del desarrollo del sistema. Al final del presente capítulo, se explica dicha selección.

2.1. Metodologías

2.1.1. Metodologías Orientadas a Objetos

La programación Orientada a Objetos ha brindado varios beneficios y facilidades a los desarrolladores, puesto que permite el uso de componentes reutilizables, bibliotecas de clases, grados de iteraciones, la división de un sistema en mini sistemas o subsistemas independientes unos de otros.



Figura 1: Partes fundamentales de las Metodologías Orientadas a Objetos.
Fuente: http://webdocs.cs.ualberta.ca/~pfiguero/soo/metod/
Elaborado Por: Ricardo Armas / Daniela Larrea

Todas las metodologías orientadas a objetos constan de partes en común como se observa en la Figura 1, siempre deben existir conceptos y diagramas para el diseño del producto, etapas a seguir durante el proceso, definición de un calendario de entregas junto con los recursos necesarios correspondiente, actividades definidas y recomendaciones sobre el desarrollo del producto.

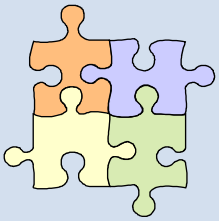
 **ASPECTOS POSITIVOS DE LAS METODOLOGÍAS ORIENTADAS A OBJETOS**
Son interactivas e incrementales. 
 Se fomenta la reutilización de componentes.
Fácil de dividir el sistema en varios subsistemas independientes. 

Tabla 1: Aspectos Positivos de las Metodologías Orientadas a Objetos

Fuente: http://es.scribd.com/doc/2050925/metodologias-de-desarrollo-software

Elaborado Por: Ricardo Armas / Daniela Larrea

Las metodologías orientadas a objetos comparten 3 aspectos positivos, expuestos en la Tabla 1, es importante el hecho de que sean interactivas ya que fomentan el trabajo en equipo, la reutilización de componentes ayuda en gran medida al ahorro de tiempo y la facilidad de dividir el sistema en pequeños módulos completamente funcionales e independientes beneficia tanto al cliente, porque obtiene lo que necesita, como también al equipo de desarrollo, por que facilita su trabajo.

Existen varias metodologías orientadas a objetos que han surgido a través de los años, como son:

- OMT, que es una metodología de análisis y diseño orientada a objetos, su popularidad radica en que tiene modalidad de abierta y no propietaria.
- Extreme Programming o XP, en español Programación Extrema se centra en el trabajo conjunto y entrega de mini versiones funcionales del sistema.
- RUP, llamada así por sus siglas en inglés Rational Unified Process, la cual consiste en la reproducción del ciclo de vida en cascada pero a menor escala y en varias iteraciones. Los Objetivos de cada iteración se establecen en función de la evaluación de las iteraciones precedentes.

2.1.2. Metodologías Ágiles

Este tipo de metodologías se centran en el desarrollo de varias versiones a lo largo de todo el ciclo de vida, el desarrollo de estas versiones incluye las fases: planificación, análisis de requerimientos, diseño, codificación y pruebas; lo importante no es la cantidad de avance o de nuevas funcionalidades que se añadan sino que existe una mejora y sin errores, al final del desarrollo de cada versión es necesario realizar una evaluación entre todos los miembros del equipo para definir prioridades para la siguiente versión.

En el proceso de desarrollo intervienen:

- Los programadores
- Directores de proyecto
- Diseñadores y

- El usuario o cliente

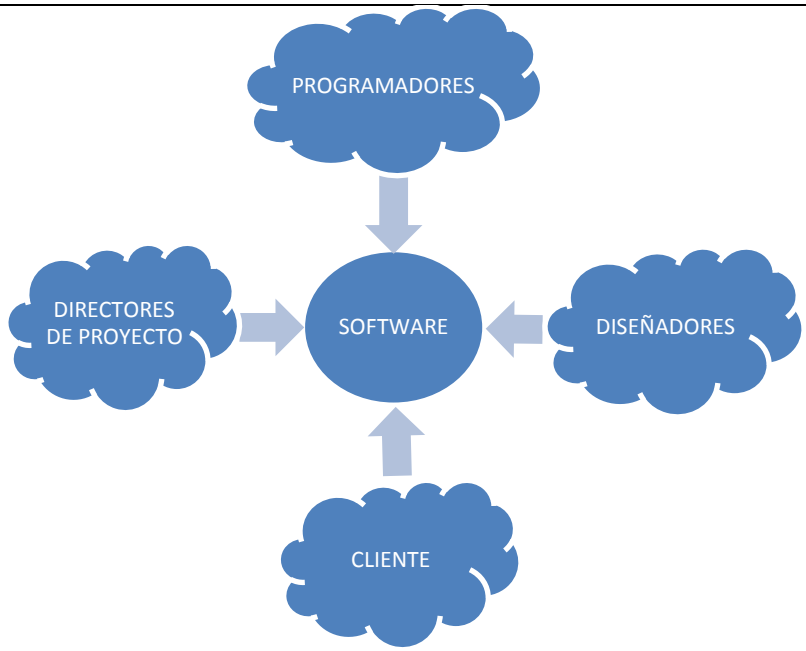


Figura 2: Involucrados en el Proceso de Desarrollo en Metodologías Ágiles
Fuente: Propia
Elaborado Por: Ricardo Armas / Daniela Larrea

Las ideas de todos los miembros son igual de valiosas, como se ve en la Figura 2, el objetivo de esta metodología es hacer que el cliente o usuario forme parte del equipo durante todo el proceso.

Beneficios

En la Tabal 2 se explica que el proceso que proponen las metodologías ágiles en comparación con otros métodos de desarrollo clásicos (ciclo de vida en cascada, evolutiva, iterativa, etc.), es menos pesado manteniendo un equilibrio donde la cantidad de pasos a seguir no retarde el proceso y todo el esfuerzo y tiempo utilizado genere resultados positivos.

Metodologías Ágiles		Metodologías Clásicas	
Adaptables	El proceso se adapta y acepta los cambios.	**Predictivas**	Se realiza un plan a largo plazo donde se rechazan los cambios.
Orientadas al equipo	Se centra en utilizar las habilidades de la gente de la mejor manera teniendo un proceso que apoye al equipo en su trabajo.	**Orientadas al proceso**	Se centran en que el proceso funcione bien sin importar los involucrados en el mismo.

Tabla 2: Metodologías Ágiles vs. Metodologías Clásicas

Fuente: Propia

Elaborado Por: Ricardo Armas / Daniela Larrea

Las metodologías ágiles están creadas y diseñadas para que todos los cambios sean bienvenidos en cualquier momento ya que el proceso mismo alienta a los miembros del equipo a responder ante estos, aún en fases tardías.

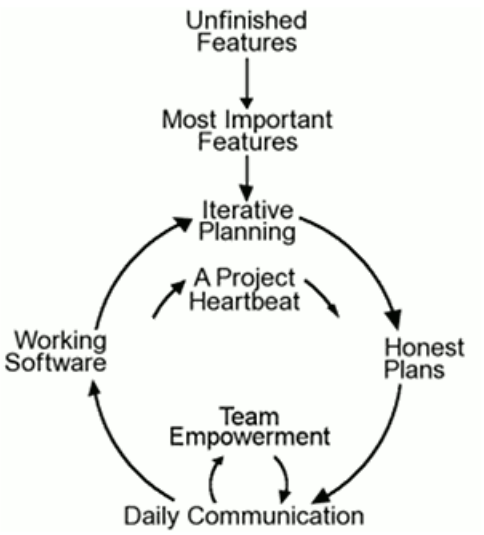


Figura 3: Metodologías Ágiles

Fuente: http://www.extremeprogramming.org/

Elaborado Por: Don Wells

En la Figura 3 se observa cómo funcionan en general las metodologías ágiles, se empieza siempre por las tareas incompletas para seguir con las de prioridad alta y poder entrar en una planeación iterativa.

Toda metodología ágil alienta el trabajo en equipo siempre teniendo en cuenta la importancia de:

- Realizar planes honestos y reales.
- La comunicación constante y de preferencia cara a cara que fortalece al equipo.
- Contar siempre con un software que funcione y corra sin problemas.
- El proyecto debe construirse con personas motivadas que tengan el soporte y confianza total para realizar su trabajo.

2.1.3. Extreme Programming

Desde sus inicios en los años 1990s, esta metodología de trabajo ha demostrado brindar beneficios significativos para todos los involucrados en el proyecto; como por ejemplo el usuario obtiene de manera muy rápida el software que requiere evitando los largos tiempos de espera, es decir el cliente recibe lo que necesita en el momento que lo necesita aumentando su satisfacción y dando la posibilidad de que surjan cambios o nuevos requerimientos.

Cada versión del software puede ser utilizado de varias maneras por el cliente, como se observa en la Tabla 3:

Uso	Beneficio
Realizar evaluaciones sobre la funcionalidad	Ayuda a determinar mejoras en tiempos de espera y diseño.
Capacitaciones al personal que manejará el software	Al realizar charlas sobre cada versión que se entrega aumentará la productividad del personal que conocerá el sistema de manera más profunda.
Implementándolo desde el principio	Ayuda a que los usuarios vean al software

para usuarios finales.	como algo tangible y no como la idea de algo que tendrán en el futuro.

Tabla 3: Uso – Beneficio de versiones entregables de Software

Fuente: http://www.extremeprogramming.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

Para la implementación del software desde el principio para usuarios finales, depende totalmente del criterio tanto del cliente que es quien evalúa las funcionalidades y sabe lo que necesita como resultado final, así como del grupo de desarrollo quienes saben el estado del software y si puede o no ofrecer las funcionalidades necesarias.

Generalmente la metodología de Extreme Programming recomienda poner al software en implementación desde las primeras versiones para que el usuario se sienta familiarizado con el sistema desde el principio.

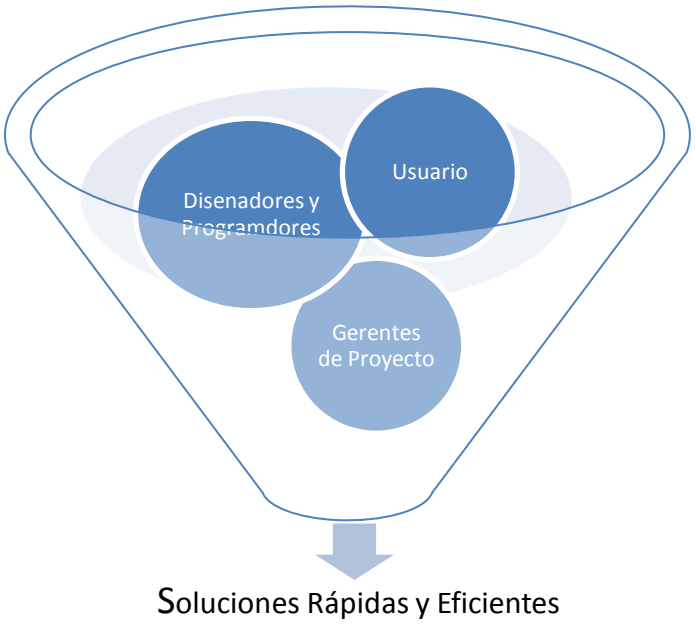


Figura 4: Involucrados equipo de desarrollo

Fuente: http://www.extremeprogramming.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

En la Figura 4 se explica que todos los miembros son iguales ya sean gerentes de proyecto, diseñadores, programadores e incluso el cliente, todos trabajan de manera colaborativa gracias al ambiente simple y práctico que permite buscar soluciones rápidas y eficientes a los problemas que se presentan.

Muchas veces los requerimientos aumentan, cambian y hasta disminuyen debido a que el usuario tiene la posibilidad de darse cuenta del efecto y esfuerzo necesario para realizar las modificaciones y por lo tanto sabrá si vale o no la pena.

El principal objetivo de esta metodología es entregar al cliente mini versiones que no tomen más de un par de semanas con funcionalidades nuevas que demuestren un avance notorio y no solo desarrollar código que sea invisible para el cliente.

Ciclo de vida de Software en Extreme Programming

En esta metodología se propone un ciclo de vida dinámico que cuenta al igual que otras metodologías con la recolección de las necesidades del usuario, realizar estimaciones, desarrollar la solución y la liberación del producto final al usuario.

La diferencia es que las etapas no se realizan una sola vez ni por largos lapsos de tiempo, sino que se las hace en tiempos cortos con en el afán de que en un par de semanas se tenga listo una versión funcional del producto para la liberación al cliente, es por ésto que el cliente tiene interacción con el sistema desde el inicio del proceso y no tiene que esperar a que se realicen todas las etapas, el momento que tiene un producto funcional en sus manos es mucho más capaz de especificar los requerimientos por que se le facilita la visualización del producto final.

En la Figura 5 se observa la diferencia del ciclo de vida de software entre Extreme Programming y dos metodologías tradicionales como son el modelo en cascada y modelos iterativos (por ejemplo incremental o espiral), claramente se observa las iteraciones que se hacen en XP para llegar al producto final.

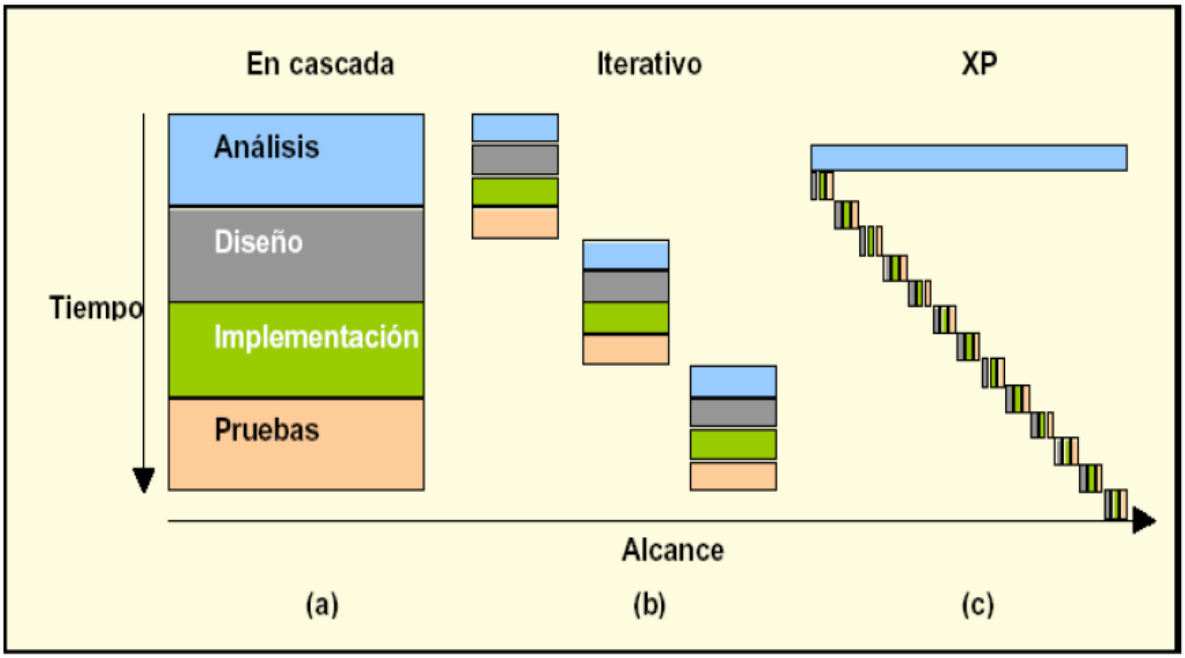


Figura 5: Tiempo – Alcance Extreme Programming vs. Metodologías tradicionales
Fuente: http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf
Elaborado Por: Ing. José Joskowicz

Se podría pensar que al tener varias iteraciones el proceso demorará más tiempo pero es todo lo contrario puesto que si bien se siguen varias etapas estas cuentan con un alcance definido y siguen reglas y prácticas lo cual previene demoras y estancamiento del proceso. Además que al empezar cada iteración se cuenta con una parte inicial completamente funcional.

Fases del ciclo de vida

A continuación se detallan las fases a seguir en cada una de las iteraciones:

- Planeación, hace referencia a todo lo relacionado con los antecedentes del proyecto es decir:
    - La recolección de los requerimientos
    - Desarrollo de cronogramas de tiempo y recursos

Antes de cada entrega de una nueva versión se debe realizar igualmente una planificación detallada donde el producto final es un Plan de Entregas o Release Plan, sobre el estado actual del sistema, las funcionalidades, los avances que ha tenido, los recursos que se encargarán de la entrega, instalación y pruebas de la versión.

- *Administración*, esta actividad es primordial en el desarrollo del software ya que brinda seguridad a todos los integrantes del grupo de trabajo asegurando un ambiente adecuado y proporcionando las herramientas necesarias para avanzar en el desarrollo evitando demoras.

- *Codificación*, el grupo de desarrolladores según sus tareas escribe el código del software organizados en pequeños grupos o más habitualmente lo hacen en parejas siguiendo la técnica de Pair Programming que pertenece a esta metodología.

En el desarrollo del código en XP se debe tener en cuenta el uso de estándares. Es muy común contar con un documento de reglas que todos los desarrolladores deben conocer y utilizar en su trabajo, generalmente se solicita el uso de comentarios y un estándar en el nombre de variables y clases dentro del código.

- *Diseño*, realizar diagramas de los módulos y funcionalidades del software que sirvan de guía para los programadores y en general para todo el grupo

de trabajo ya que al ser en forma gráfica hace más simple la comprensión para todos. También facilita la revisión de posibles errores, de faltantes o de nuevas funcionalidades que se puedan añadir al sistema en versiones posteriores. En XP es necesario cumplir con los siguientes criterios de diseño:

Simplicidad	Soluciones "spike"	Recodificación	Metáforas
•La Implementación es mas rápida con diseños simples que con diseños complejos.	•Spike se refiere a programas de prueba que ayudan a explorar soluciones a problemas técnicos. • Generalmente son desechados cuando han terminado de cumplir su misión.	•Escribir nuevamente una fracción del código sin cambiar su funcionalidad pero haciendolo más conciso y entendible.	•Utilizar metáforas que sin necesidad de mayor explicación son entendidas. •Una buena metáfora tiene un contenido suficiente y es bien entendida por todo el equipo.

Tabla 4: Criterios de Diseño

Fuente: http://www.extremeprogramming.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

El uso de metáforas expuestas en la Tabla 4, se refiere a una descripción de cómo funciona el programa en una visión global.

- *Pruebas*, la detección de errores es muy importante en el desarrollo del software porque cuando se llega al origen del error se pueden detectar otros errores o encontrar la solución a errores similares.

En la Tabla 5 se detalla el plan de pruebas, en donde las pruebas que se hacen son originadas por los propios errores, requerimientos o estándares a seguir. Generalmente la persona que fue capaz de detectar el error es la más indicada

para buscar posibles soluciones. El costo que representan las pruebas han llevado a pensar que entre menos pruebas se haga es mejor por los recursos y tiempo que se requiere, en Extreme Programming se cuenta con un plan de pruebas definido como se observa a continuación:

PLAN DE PRUEBAS	
Pruebas Unitarias	Antes de empezar el desarrollo de la nueva versión se realiza un plan de pruebas unitarias que realizan los desarrolladores, por lo tanto se cuenta con las mínimas pruebas a pasar antes de escribir el código.
Detección y corrección de errores	Al detectar un error, se resuelve inmediatamente para evitar errores similares, o el efecto del error en otras partes del módulo, o del producto en general.
Pruebas de aceptación	Son consideradas como pruebas de caja negra en donde el cliente se encarga de especificar casos de prueba y de aprobar el correcto funcionamiento del sistema.

Tabla 5: Plan de Pruebas

Fuente: http://www.extremeprogramming.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

Reglas para el ciclo de vida

Las reglas para cada actividad son muy claras y al seguirlas al pie de la letra se puede en realidad ver los beneficios de la metodología. Extreme Programming enfatiza que cada miembro del grupo además de cumplir con sus tareas debe tener en mente las siguientes reglas para el desarrollo de software.

Para la Planificación las reglas se detallan en la Tabla 6:

REGLAS EN PLANIFICACIÓN	
① *ESCRIBIR LAS "HISTORIAS DE USUARIOS" O "USER´S STORIES"*	• Las historias de usuario son escritos cortos del usuario con el mismo propósito de los diagramas de casos de uso. • Sirven para reemplazar el documento de requerimientos por lo tanto no se limitan solo a describir los módulos del sistema sino los usuarios escriben en su propia terminología, es decir, en lenguaje no técnico, sobre lo que necesitan que el sistema haga en su beneficio.
② *CREAR EL PLAN DE ENTREGAS QUE GENERA EL CALENDARIO DE ENTREGAS.*	• La toma de decisiones se divide entre todos los miembros del equipo pero cada uno toma decisiones en su rama. • El plan se elabora en base de las historias de usuarios donde se establece prioridades y tiempos. • Para poder decir que se tiene un plan final es necesario negociar hasta que todos los miembros estén de acuerdo.
③	• Un proyecto se puede calificar usando 4 variables, que son: TIEMPO, CALIDAD, RECURSOS y ALCANCE.

SEGUIR LA FILOSOFÍA DE LAS 4 VARIABLES	• Una sola persona no puede controlar las 4 variables al mismo tiempo, porque el cambio de una variable tiene un impacto directo en las otras 3. • Es necesario repartir el control, por ejemplo los desarrolladores pueden encargarse de 1 y los gerentes de proyecto ocuparse de las otras 3.
④ ***EL PROYECTO ESTÁ DIVIDO EN ITERACIONES.***	• En cada iteración se realiza una planificación y al final se cuenta con una versión nueva y funcional del software que es entregada al cliente quien decide si se la pone o no en producción. • Cada iteración debe durar de 1 a 3 semanas y el esfuerzo debe estar centrado en completar con las tareas definidas en el plan de entregables.

Tabla 6: Reglas de la Planificación

Fuente: http://www.extremeprogramming.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

Según Ron Jeffries en su portal http://xprogramming.com/what-is-extreme-programming , las iteraciones son realizadas de 2 maneras importantes:

Primero, el equipo libera un software probado y funcional con lo requerido. El cliente decide si lo usa para evaluación o lo pone en producción para usuario finales (muy recomendable). Lo más importantes es mantener al software y a los avances visibles al final de cada iteración.

Segundo, el equipo libera el software frecuentemente a sus usuarios finales. Los proyectos XP en Web realizan liberaciones diarias y los proyectos in house liberaciones mensuales o más frecuentes.

REGLAS EN ADMINISTRACIÓN	

① ***REALIZAR REUNIONES DIARIAS***	• En estas reuniones se busca discutir sobre problemas, soluciones y promover la visión del proyecto.
	• Son reuniones cortas donde deben asistir todos los miembros del equipo, evitando así reuniones largas o numerosas solo con los involucrados en cada caso.
	• Los desarrolladores deben reportar 3 cosas importantes:
	  ○ Cuál fue su avance del día anterior
	  ○ Las causas de retrasos
	  ○ Cuál es su avance esperado para el día actual.

② ***MEDIR LA VELOCIDAD DEL PROYECTO***	• La velocidad o rapidez del proyecto se refiere a cuantas tareas están siendo desarrolladas con respecto a las historias de usuarios que se tienen. • También se toma en cuenta el total de las tareas finalizadas por cada iteración terminada. • Es normal que la velocidad varíe y se puede presentar una variación en más de una iteración.
③ ***ROTAR A LOS MIEMBROS DEL EQUIPO EN DIFERENTES RESPONSABILIDADES.***	• Para evitar tener expertos en una sola rama o actividad es necesario rotar a las personas, esto libera la dependencia del proyecto en alguna determinada persona. • El equipo se vuelve más flexible y eficiente si todos tienen un conocimiento general de todos los módulos del sistema.

Tabla 7: Reglas de la Administración
Fuente: http://www.extremeprogramming.org/
Elaborado Por: Ricardo Armas / Daniela Larrea

Además de las reglas expuestas en la tabla 7, en la fase de Administración es importante tener un control sobre cómo está siendo utilizada la metodología XP por parte de los miembros del equipo, ya que se podría requerir de cambios, de una reestructuración o de motivación extra al equipo de trabajo.

Para la fase de Diseño, a parte de los criterios necesarios ya mencionados anteriormente se tiene 2 reglas fundamentales como se observa en la Tabla 8:

REGLAS EN DISEÑO	
① ***UTILIZAR LAS "CRC CARDS"***	• Las CRC Cards (Class, Responsibilities and Collaboration Cards) permiten que todo el equipo contribuya con ideas para el diseño. • Cada cartas se utilizan para representar un objeto por lo tanto se puede tener una idea más clara de cómo interactúan los objetos entre sí y así el diseño final será mucho más fácil y todos los miembros estarán de acuerdo con el mismo.
② ***NUNCA AÑADIR FUNCIONALIDADES EXTRAS***	• No se debe añadir al sistema funcionalidades extra sin que sean necesarias o antes de lo planificado. • Nunca se puede adivinar lo que el cliente necesita y es mejor evitar la pérdida de tiempo y el preocuparnos por otra cosa que no sea lo que debe hacerse y lo que está planificado.

Tabla 8: Reglas en Diseño
Fuente: http://www.extremeprogramming.org/
Elaborado Por: Ricardo Armas / Daniela Larrea

Así mismo, para la fase de codificación, será importante seguir las cuatro reglas ilustradas en la Tabla 9.

REGLAS EN CODIFICACIÓN	
① ***RITMO SOSTENIBLE***	• Mantener un buen ritmo sin sobrecargar a nadie, evitar el trabajo excesivo fuera de horario normal porque resulta en desmotivación y por lo tanto la calidad decae.
② ***USO DE ESTANDARES***	• Como otras metodologías, XP propone el uso de estándares para contar con un código entendible y de fácil modificación.
③ ***PROPIEDAD COLECTIVA DEL CÓDIGO***	• Todos son dueños de todos los módulos y cualquiera puede contribuir con ideas no solo para la parte que se encuentre desarrollando sino en general.
④ ***INTEGRACIONES PERMANENTES***	• Es importante contar con un repositorio de fuentes actualizado y libre de errores para que todos tengan las últimas versiones de todo el producto. • Controlar de manera oportuna los módulos y las partes en que cada desarrollador o pareja de desarrolladores están trabajando para evitar cruzarse o dañar el trabajo de los demás.

Tabla 9: Reglas en Codificación

Fuente: http://www.extremeprogramming.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

A continuación en la Tabla 10, se tienen las tres reglas principales para llevar a cabo la etapa de pruebas.

REGLAS EN PRUEBAS	
① ***TODO EL CÓDIGO DEBE APROBAR LAS PRUEBAS UNITARIAS ANTES DE SU LIBERACIÓN***	• Las pruebas deben ser creadas antes del código. Si se encuentra o se crea una nueva prueba se la debe ejecutar antes de la liberación del código. • Entre más difícil sea describir y crear las pruebas son más necesarias para el proyecto.
② ***CUANDO SE ENCUENTRA UN ERROR SE AUMENTAN PRUEBAS***	• Por cada error encontrado se crea una prueba en contra del mismo, esto evita tener errores del mismo tipo en el futuro. • Cuando se encuentra un error en producción se debe crear primero una prueba de aceptación la cual alerta a los desarrolladores quienes a su vez crean una prueba unitaria la cual al ser aprobada se asegura que no habrá problema de pasar por la prueba de aceptación posteriormente.
③ ***LAS PRUEBAS DE ACEPTACIÓN DEBEN SER REALIZADAS FRECUENTEMENTE***	• Las pruebas de aceptación son creadas en base a las historias de usuarios y son básicamente casos de prueba definidos por el propio usuario final. • Estas pruebas generan una calificación la cual debe ser analizada por los desarrolladores para saber si cumple o no con el resultado esperado. • Es necesario contar con un grupo de QA (Quality Assurance) que se encuentra integrado al grupo de desarrolladores y se encarga de asegurar el paso correcto de las pruebas de aceptación.

Tabla 10: Reglas en Pruebas

Fuente: http://www.extremeprogramming.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

Extreme Programming no tiene el objetivo de ser un conjunto de reglas estrictas a seguir, sino es una forma de trabajo que se ajusta a los valores personales y corporativos de los equipos de trabajo. Por lo tanto, se tiene la libertad de tomar la decisión de seguir todas las reglas, cambiar algunas o ajustarlas a las necesidades del proyecto y esto se puede hacer desde el inicio o ir realizando cambios paulatinos para cada iteración con el conocimiento y experiencia adquiridos en iteraciones anteriores.

Técnica Pair Programming

La programación en parejas consiste principalmente en que todo el código que será enviado a producción cuando esté terminado, sea creado por dos personas trabajando juntas en la misma máquina.

Según la Organización Extreme Programming Org., la programación en parejas incrementa la calidad del producto sin impactos en el tiempo de entrega. Así mismo, señala que la mejor manera de aplicar esta técnica es sentándose el uno al lado del otro en frente del monitor. De esta manera, los dos están siempre atentos y analizando el código, para que cualquiera intervenga cuando tenga que hacerlo.

Algo muy importante que se debe tener siempre en cuenta durante todo el desarrollo aplicando Pair Programming es el nivel de los 2 desarrolladores, no debe existir una gran diferencia entre ellos, es mejor cuando ambos están más o menos al mismo nivel debido a que trabajan conjuntamente intercambiando posiciones, corrigiéndose y discutiendo sobre las posibles soluciones. Para un trabajo exitoso se debe contar siempre con un cronograma, donde las horas de trabajo sean suficientes para terminar las tareas pero mantener un ritmo constante sin tener días de trabajo excesivo ni días con muy pocas horas de trabajo.

Aplicando esta técnica puede resultar complicado acostumbrarse a la sinceridad total con la pareja de programación, se proponen algunas ideas a poner en práctica:



Figura 6: Pair Programming
Fuente: http://www.extremeprogramming.org/
Elaborado Por: Ricardo Armas / Daniela Larrea

En la Figura 6, se observan a 2 programadores practicando la técnica de pair programming junto con las frases recomendadas para las situaciones más comunes. A continuación en la Tabla 11 una breve explicación sobre cada una de las frases:

①	Puede ser que las dos personas tengan una buena idea pero generalmente, los mismos tratan de imponer su idea entonces la recomendación es ceder y probar la idea del otro aunque no se esté de acuerdo.
②	Siempre se debe tener en cuenta que el trabajo es de las dos personas, entonces el que está en frente del computador debe mantener un ritmo en la escritura del código para evitar que la otra persona se pierda.
③	Es común que al escribir el código exista una concentración tal que se pueden olvidar las reglas, pero para evitar esto una de las dos personas debe siempre estar pendiente del cumplimiento de las reglas.
④	Muchas veces pueden existir mal entendidos entre los desarrolladores, se deben probar con diferentes formas de explicarse como dibujando diagramas alejándose completamente del código.
⑤	Es común cometer errores de escritura por lo tanto el que escribe el código se encarga de todos los detalles y la otra persona debe poner más atención en el contexto general y no fijarse en cada una de las letras que el otro esté escribiendo.

Tabla 11: Pair Programming (Detalle)

Fuente: http://www.extremeprogramming.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

Suele ser difícil comunicarse con total sinceridad, sobre todo cuando se discute sobre algún tema y se tiene diferentes puntos de vista pero siempre tiene que reinar el respeto y la paciencia. No es necesario que una de las dos personas someta a la otra sino que se busca la mejor manera de explicar las ideas, hacer pequeñas pruebas para que el que esté equivocado entienda y se dé cuenta de sus errores.

Variaciones de Pair Programming

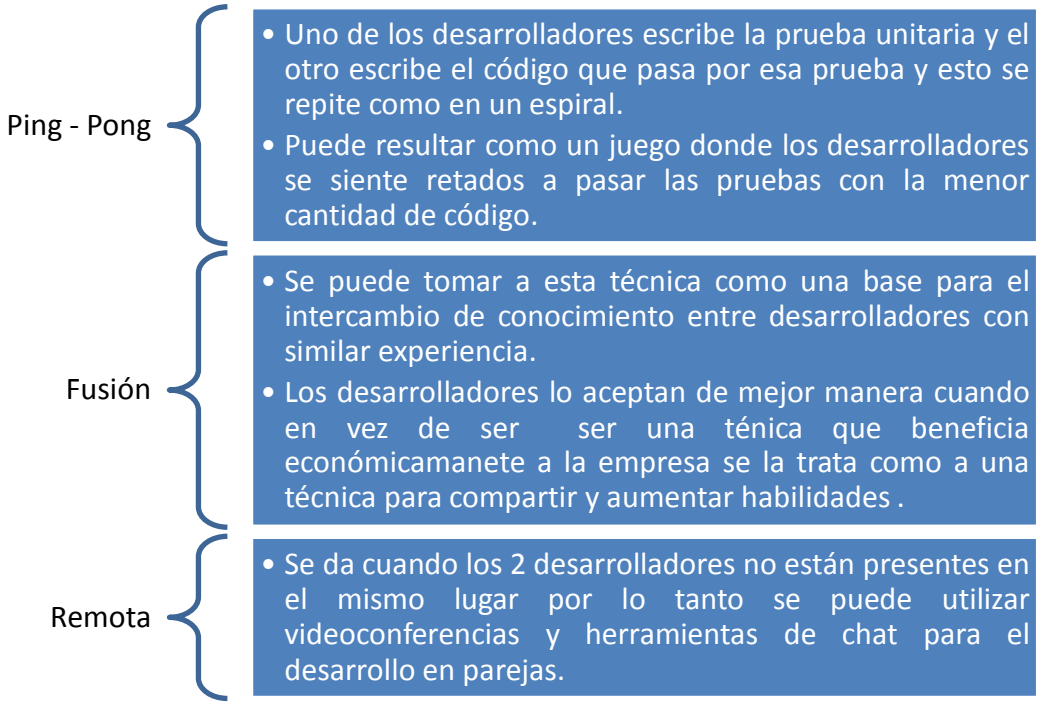


Tabla 12: Variaciones de Pair Programming

Fuente: http://www.d80.co.uk/post/2011/02/18/Extreme-Pair-Programming.aspx

Elaborado Por: Ricardo Armas / Daniela Larrea

Ventajas de Pair Programming

A continuación en la Figura 7 se observan las ventajas de la técnica:

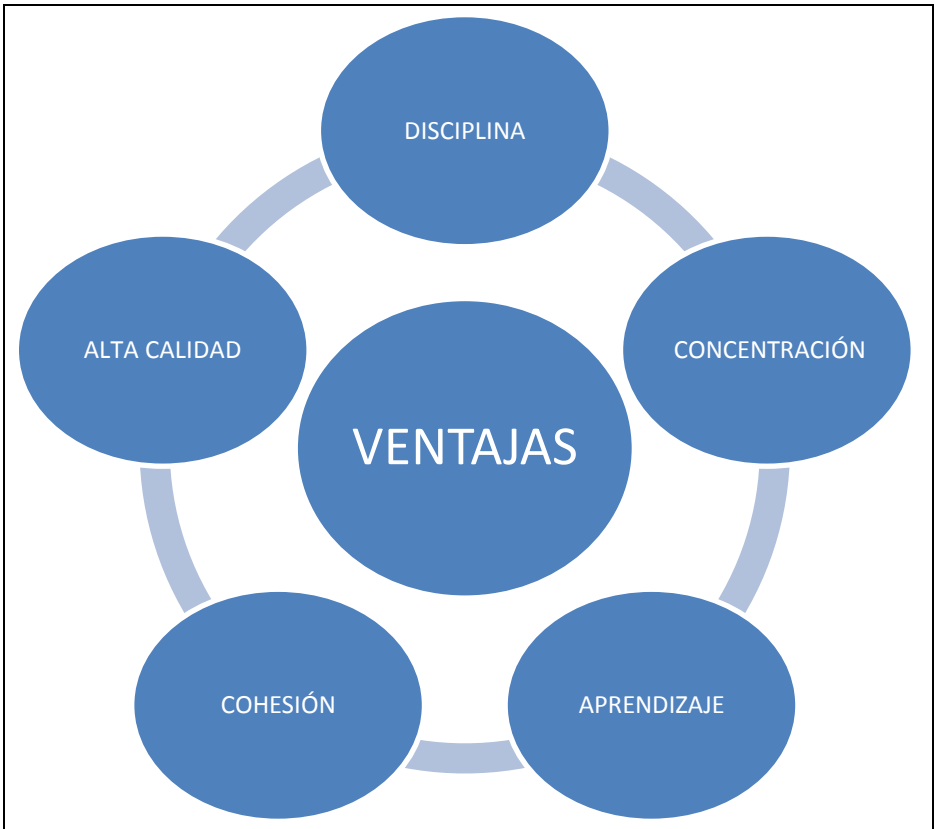


Figura 7: Ventajas de Pair Programming

Fuente: http://www.extremeprogramming.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

- Es evidente que si se está trabajando entre dos desarrolladores se va a aprovechar el tiempo en lo que se tiene qué hacer y por lo tanto las distracciones son casi nulas y se logra mantener un flujo constante de trabajo

- La relación entre los miembros del equipo de desarrollo se ve positivamente transformada y mejora notablemente, puesto que al trabajar tan

continuamente con una persona se llega a tener más confianza y por ende el ambiente de trabajo se vuelve más agradable.

- Para logar un éxito con la programación en parejas lo mejor es que las parejas no sean fijas, que roten entre sí, así como también se roten los roles de las personas, así se logra que todo el equipo tenga un conocimiento de todo el código en general.

- A pesar de que la programación en parejas no es una tutoría ni una relación profesor – alumno, se puede adquirir nuevos conocimientos y habilidades diariamente. Mientras las personas se sientan comprometidas con este tipo de trabajo se verá beneficiado tanto el proyecto como el equipo.

- Existe el malentendido de que utilizar esta técnica resulta más costoso porque "Dos personas hacen el trabajo que podría realizar una sola", pero en realidad es todo lo contrario porque la calidad es mejor, el código es mejor, las tareas son completadas en menor tiempo y por lo tanto se disminuyen los retrasos.

Involucrados

En la figura 8 se observa que en esta técnica existen dos personas involucradas denominados: conductor y navegador; cada uno con su rol definido, aunque como ya se dijo es importante que se intercambien los roles cada cierto tiempo puede ser cada 2 o 3 horas para lograr mejores resultados:

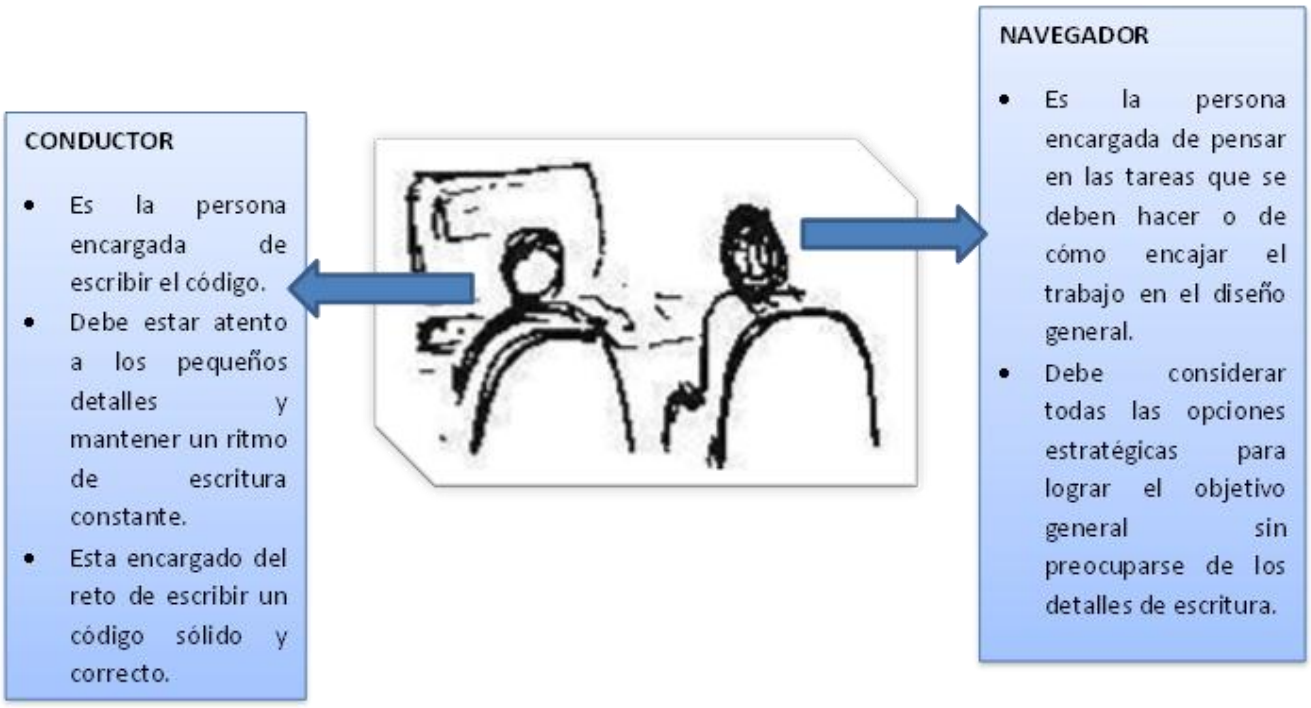


Figura 8: Involucrados en Pair Programming

Fuente: http://www.extremeprogramming.org/
Elaborado Por: Ricardo Armas / Daniela Larrea

Es importante que cada uno desempeñe sus responsabilidades de la mejor manera, el navegador no debe caer en el error de corregir los errores de escritura del conductor a cada momento porque se vuelve una situación molestosa y se puede dejar a un lado el objetivo real al preocuparse más de pequeños detalles. El navegador debe significar una ayuda y apoyo al conductor no una molestia ni una fuente de presión negativa.

2.2. Herramientas

A continuación se puede observar en la Tabla 15 las herramientas a utilizar para el desarrollo de un sistema de software.

Herramienta	Descripción
Lenguajes de Modelamiento UML	Lenguajes para describir procesos, flujos y actividades que realiza un sistema orientado a objetos.
Lenguajes de Programación	Herramientas que permiten crear programas y software.
Base de Datos	Herramienta para almacenar datos, organizados en tablas.

Tabla 15: Herramientas para desarrollo de un sistema
Fuente: Propia
Elaborado Por: Ricardo Armas / Daniela Larrea

2.2.1. Lenguajes de Modelamiento UML

"El Lenguaje Unificado de Modelado consiste en un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos y describe el significado esencial de estos diagramas y símbolos utilizados en ellos." (María Jesús Lamarca Lapuente, Hipertexto, el nuevo concepto de documento en la cultura de la imagen, Tesis doctoral. Universidad Complutense de Madrid, 10/12/2009)

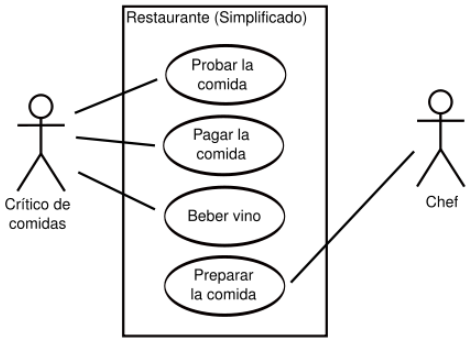
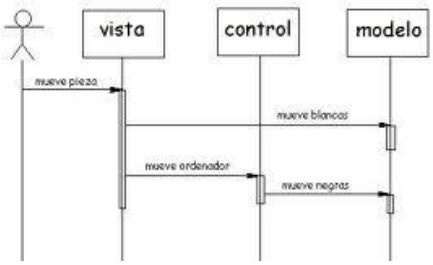
Diagrama	Descripción	Ejemplo
Diagrama de Casos de Uso	Se utilizan para modelar los procesos de las reglas del negocio. Intervienen los actores, que son las personas que estarán en interacción con el sistema, y los procesos, que son las acciones que los actores van a realizar con ayuda del sistema.	
Diagrama de Secuencia	Se utilizan para modelar el paso de mensajes entre objetos. Describen el orden en el que se van a dar las actividades en el tiempo dentro de cada proceso e indican a que proceso pertenecen.	
	Se utilizan para modelar interacciones	

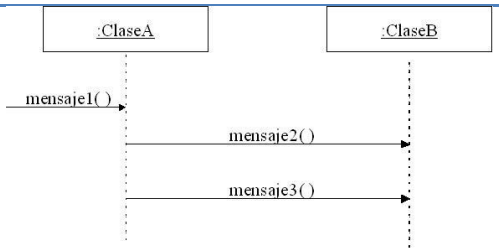
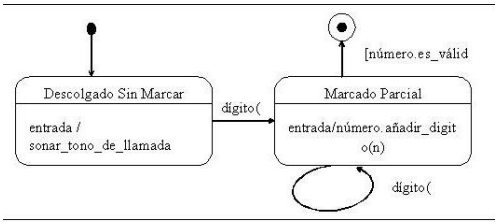
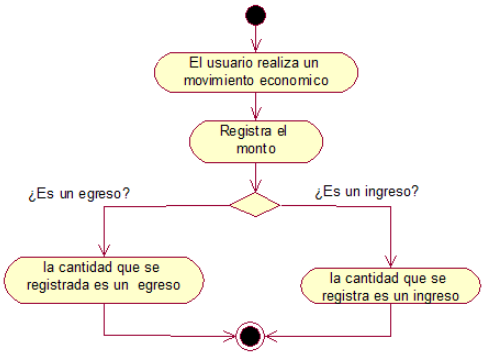
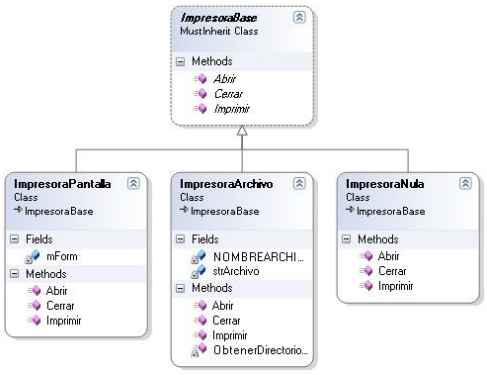
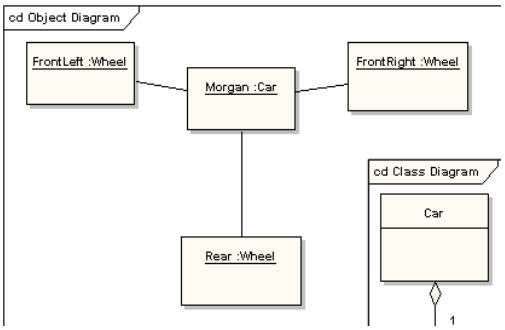
Diagrama de Colaboración	entre objetos. Pueden mostrar el contexto de la operación, es decir, cuales objetos son atributos, cuales son temporales, etc.	
Diagrama de Estado	Se utilizan para modelar el comportamiento de los objetos en el sistema. En éste, se indican qué eventos hacen que se pase de un estado a otro y cuáles son las respuestas y acciones que genera.	
Diagramas de Actividad	Se utilizan para modelar el comportamiento de los Casos de Uso, objetos u operaciones. Estos diagramas muestran el flujo de trabajo desde el punto de inicio hasta el punto final detallando las rutas y decisiones tomadas a lo largo de la actividad.	

Diagrama de Clases	Se utilizan para modelar la estructura estática de las clases en el sistema. Se describen los atributos y funciones que cada una presenta, y también la relación que tienen entre sí.	
Diagrama de Objetos	Se utilizan para modelar la estructura estática de los objetos en el sistema. Utilizan un sub conjunto de elementos de un diagrama de clase para enfatizar la relación entre las instancias de las clases en algún punto en el tiempo.	

Diagramas de Componentes

Se utilizan para modelar los componentes del sistema.

Los componentes representan todos los tipos de elementos de software que entran en la fabricación del sistema.

Estos elementos pueden ser simples archivos, paquetes, bibliotecas, etc.

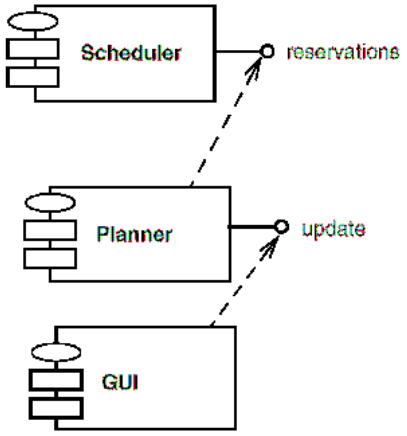
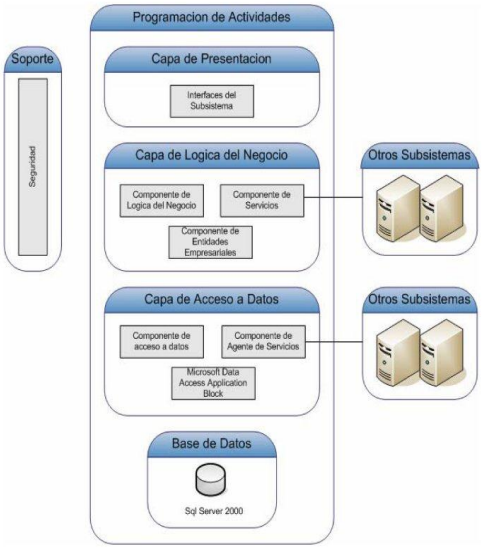


Diagrama de Implementación

Se utilizan para modelar la distribución del sistema.

Muestran las instancias existentes al ejecutarse así como sus relaciones.

También se representan los nodos



	que identifican recursos físicos, típicamente un ordenador así como interfaces y objetos

Tabla 16: Tipos de Diagramas para modelar Sistemas

Fuente: http://www.hipertexto.info/documentos/uml.htm, 2011

Elaborado Por: Ricardo Armas / Daniela Larrea

2.3. Lenguajes de Programación

Los sistemas operativos aparecieron a principios de 1950 con el objetivo de facilitar la interacción entre las personas y las computadoras. En estas primeras máquinas la programación se hacía manualmente conectando cables y pulsando interruptores, y tomaban alrededor de media hora hasta un día entero para realizar cálculos simples.

Lenguaje De Máquina: No mucho después, nace el famoso Lenguaje de Máquina, que consistía en programar en base a instrucciones, que eran secuencias de unos y ceros. Y a principios de los 50's se desarrollaron los primeros lenguajes simbólicos que se dieron por necesidad de utilizar secuencias de programación prediseñadas para las acciones usuales como sumas, restas, multiplicaciones, etc. Este conjunto de instrucciones fue denominado Lenguaje Ensamblador.

Los lenguajes de programación son herramientas que permiten crear programas y software. Estos facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas.

Existen lenguajes de programación que utilizan un compilador y la ejecución de un programa con compilador requiere de dos etapas:

- Traducir el programa simbólico a código máquina

- Ejecución y procesamiento de los datos.

Otros lenguajes de programación utilizan un programa intérprete o traductor, el cual analiza directamente la descripción simbólica del programa fuente y realiza las instrucciones dadas. El intérprete en los lenguajes de programación simula una máquina virtual, donde el lenguaje de máquina es similar al lenguaje fuente, por eso la ventaja del proceso interprete es que no necesita de dos fases para ejecutar el programa, sin embargo su inconveniente es que la velocidad de

ejecución es más lenta ya que debe analizar e interpretar las instrucciones contenidas en el programa fuente al mismo tiempo.

Lenguaje	Principal área de aplicación	Compilado/interpretado
ADA	Tiempo real	Lenguaje compilado
BASIC	Programación para fines educativos	Lenguaje interpretado
C	Programación de sistema	Lenguaje compilado
C++	Programación de sistema orientado a objeto	Lenguaje compilado
Cobol	Administración	Lenguaje compilado
Fortran	Cálculo	Lenguaje compilado
Java	Programación orientada a Internet	Lenguaje intermediario
MATLAB	Cálculos matemáticos	Lenguaje interpretado
Cálculos matemáticos	Cálculos matemáticos	Lenguaje interpretado
LISP	Inteligencia artificial	Lenguaje intermediario
Pascal	Educación	Lenguaje compilado
PHP	Desarrollo de sitios web dinámicos	Lenguaje interpretado
Inteligencia artificial	Inteligencia artificial	Lenguaje interpretado
Perl	Procesamiento de cadenas de caracteres	Lenguaje interpretado

Tabla 17: Tipos de Lenguajes de Programación

Fuente:
http://2.bp.blogspot.com/_RoP0c8KoLrg/TK1R8K4uh2I/AAAAAAAAADM/oOVkPzEqvLc/s1600/LENG.jpg , 2011

Elaborado Por: ELISEO Y ROSA MARIA (INSALHER), Octubre 2010.

Como se puede ver en la Tabla 17, se tiene toda una gama de lenguajes tanto interpretados como compilados que se pueden utilizar como herramienta de desarrollo de un sistema. En la lista anterior se muestran los lenguajes más comunes que se han venido utilizando por programadores durante años.

2.3.1. Programación Orientada a Objetos

La programación orientada a objetos es una de las formas más populares de programar y viene teniendo gran acogida en el desarrollo de proyectos de software en los últimos años. Esta acogida se debe a sus capacidades y ventajas frente a las antiguas formas de programar.

Como se conoce, la programación fue hecha en una manera secuencial o lineal, es decir, siguiendo una serie de pasos consecutivos con estructuras consecutivas.

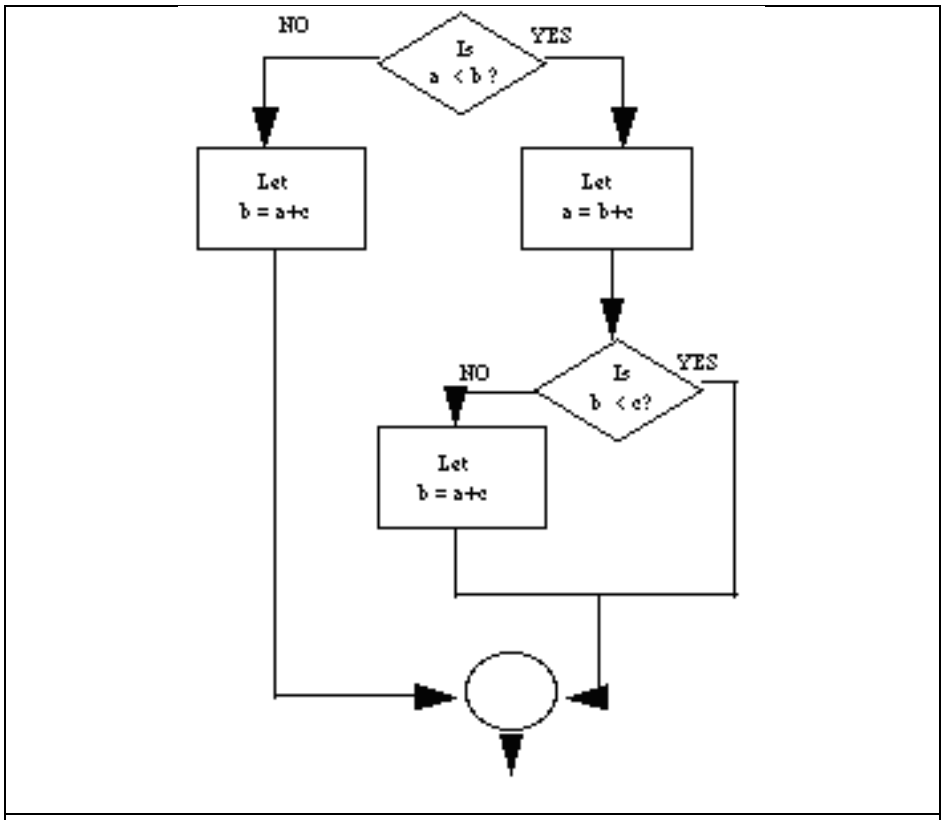


Figura 9: Programación – Perspectiva Histórica

Fuente: http://java.ciberaula.com/articulo/tecnologia_orientada_objetos/

Elaborado Por: © 2010 - Todos los derechos reservados Ciberaula España – USA – México - Colombia - Chile - Argentina

Los lenguajes basados en esta forma de programar, ofrecieron ventajas de orden y organización en un principio, pero a medida que los sistemas se iban haciendo más complejos, esta programación no ofrecía flexibilidad ni dinamismo, y mantener grandes cantidades de código en un mismo bloque se volvía cada vez más complicado.

Frente a todo esto, fueron apareciendo los lenguajes basados en la programación estructurada, en los cuales la idea principal consiste en separar las partes complejas del programa en módulos, de esta manera, se puede ir llamando al módulo que se necesite cuando se lo necesite. Poco a poco esta nueva forma de programar fue remplazando a la conocida programación lineal.

A medida que la tendencia a crear programas mejores, más complejos y más grandes fue aumentando, los programadores comenzaron a buscar y desarrollar nuevas formas de programar, creando como producto la Programación Orientada a Objetos (POO). La POO viene de la evolución de la programación estructurada. Se basa en dividir el programa en pequeñas unidades lógicas de código, a los que se denomina Objetos. Estos objetos se comunican entre sí utilizando mensajes a lo largo de todo el programa.

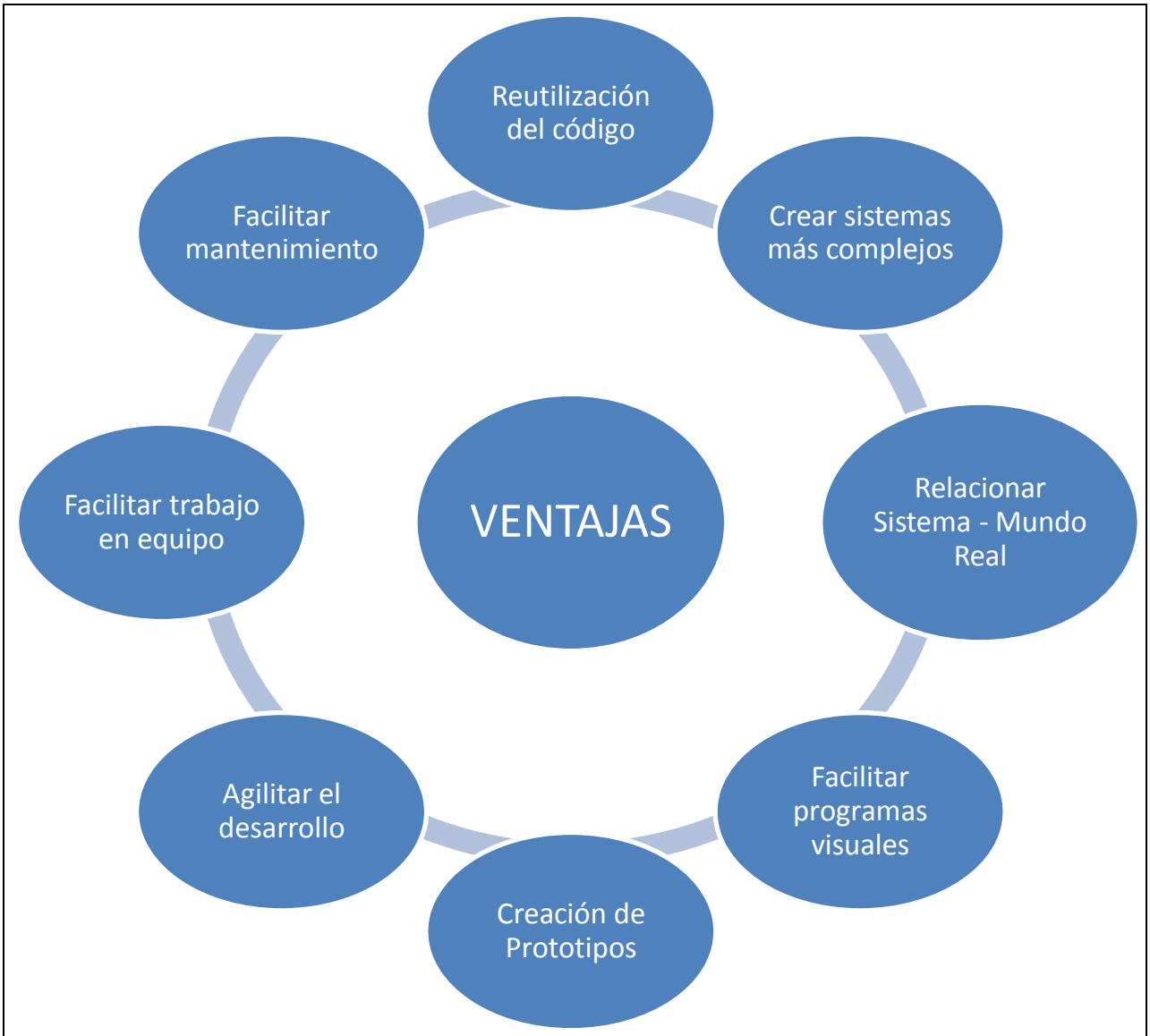


Figura 10: Ventajas de la Programación Orientada a Objetos

Fuente: Propia

Elaborado Por: Ricardo Armas / Daniela Larrea

Ventajas de la Programación Orientada a Objetos

Las principales ventajas de la programación orientada a objetos que se ilustran en la Figura 10, se las revisa a continuación:

- Sistema <-> Mundo Real: La POO utiliza herramientas que ayudan a modelar y representar el mundo real. En el mundo real existen objetos, que presentan características y un comportamiento definido. Por ejemplo, los automóviles tienen características (marca, modelo, color, velocidad máxima, etc.) y comportamiento (frenar, acelerar, retroceder, llenar combustible, cambiar llantas, etc.). Los Objetos de Software, al igual que los objetos del mundo real, presentan los mismos componentes, excepto que mantiene sus características en una o más variables, y su comportamiento en métodos. Un método es una función asociada al objeto.

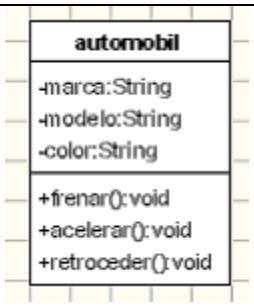


Figura 11: Objeto

Fuente: http://java.ciberaula.com/articulo/tecnologia_orientada_objetos/

Elaborado Por: Ricardo Armas / Daniela Larrea

- Reutilización Del Código: Al trabajar con objetos, se puede aprovechar el hecho de que ya quedaron definidos con sus variables y métodos, por lo tanto, reutilizarlos a lo largo de todo el programa con solo instanciarlos cuando sea necesario.

- Facilidad En Mantenimiento De Software: Los objetos, al mantener su esencia (variables y métodos) a lo largo del programa, facilitan el proceso de mantenimiento de software, ya que si se llega a dar el caso en que se tiene que cambiar alguna de sus características, añadir otra o quitar alguna, sólo

se lo tiene que hacer en la definición del mismo y esto se reflejará a lo largo de todo el programa.

- Trabajo En Equipo: Gracias a la característica mencionada en el Mantenimiento de Software, los objetos se muestran globales al programador y al equipo. O sea, facilitan el trabajo en equipo, por el hecho de que no se los tiene que volver a definir y se reconocen fácilmente, así el programador no haya tenido interacción previa con ellos.

- Agilitar el Desarrollo: Es una de las ventajas de la Programación Orientada a Objetos, y se debe a su enfoque y orientación a trabajar en base a objetos. El hecho de definirlos una sola vez en todo el programa, poder volver a utilizarlos cuando sea necesario, y que cualquier otro miembro del equipo lo haga, logran la agilidad mencionada.

2.3.2. Lenguaje de Programación JAVA

Según expone Sun, java es: "Un lenguaje simple. Orientado al objeto, distribuido, interpretado, sólido, seguro, de arquitectura neutral, portable, de alto desempeño, de multihilos y dinámico" (http://java.sun.com, Sept. 2011).

En el siguiente cuadro de Características de Java como Lenguaje de Programación, se puede observar cómo se describe cada una de las definiciones proporcionadas en la cita anterior por Sun.

JAVA	
① **SIMPLE**	Java es basado en el lenguaje C++, pero con diferencias que remarcan la cualidad de SIMPLE: • Java no da soporte a struct, union y pointer. • Java no ofrece typedef ni #define.

	• Java no permite la sobrecarga de operadores. • Java no ofrece herencia múltiple. • Java tiene una clase String, que permite un mejor manejo que los arrays de terminación nula del C y C++. • Java tiene un sistema automático de asignación y liberación de memoria (recolector de basura) que mejora mucho los sistemas del C++
② ORIENTADO AL OBJETO	Java soporta su esencia de ser un lenguaje Orientado a Objetos, todas sus técnicas, y en resumen a la reutilización de componentes de software.
③ DISTRIBUIDO	• Java fue diseñado para trabajar en ambiente de redes y contienen una gran biblioteca de clases para la utilización del protocolo TCP/IP, incluyendo HTTP y FTP. • El código Java se puede manipular a través de recursos URL con la misma facilidad que C y C++ utilizan recursos locales (archivos).
④ INTERPRETADO	El compilador Java traduce cada fichero fuente de clases a código de bytes, que puede ser interpretado por todas las máquinas que den soporte a un visualizador que funcione con Java.
⑤ SÓLIDO	• El código Java no se quiebra fácilmente ante errores de programación. • No es posible la conversión forzada (cast) de enteros en punteros y no ofrece soporte a los punteros que permitan saltarse reglas de manejo de tipos. • No es posible escribir en áreas arbitrarias de memoria ni realizar operaciones que corrompan el código.

⑥ **SEGURO**	Debido a que Java suele funcionar en ambiente de redes, el tema de seguridad interesa en sobremanera. Las mismas características antes descritas que evitan la corrupción de código evitan su manipulación.
⑦ **ARQUITECTURA NEUTRAL**	El compilador crea códigos de byte que puede ser interpretado en cualquier máquina Virtual Java.
⑧ **PORTABLE**	• Al ser de arquitectura neutral es altamente portable, la máquina virtual de Java es multiplataforma. • Los tipos de dato estándares (int, float, double) están igualmente implementados en todas las máquinas por lo que las operaciones aritméticas funcionaran igual sin importar la plataforma.
⑨ **ALTO DESEMPEÑO**	El intérprete Java puede ejecutar directamente el código.
⑩ **MULTIHILOS**	Java puede aplicarse a la realización de aplicaciones en las que ocurra más de una cosa a la vez. Apoyándose en un sistema de gestión de eventos permite apoyar la conducta en tiempo real e interactiva en programas
⑪ **DINÁMICO**	Al contrario que C++ que exige se compile de nuevo la aplicación al cambiar una clase madre, Java utiliza un sistema de interfaces que permite aligerar esta dependencia. Como resultado, los programas Java pueden permitir nuevos métodos y variables en un objeto de biblioteca sin afectar a los objetos dependientes.

Tabla 18: Características de Java como Lenguaje de Programación

Fuente: http://www.infor.uva.es/~jmrr/2007/TGP/DOC/JAVA.html

Elaborado Por: Ricardo Armas / Daniela Larrea

2.4. Bases de Datos

Una base de datos es un conjunto de datos pertenecientes al mismo contexto almacenados sistemáticamente para su posterior uso. La informática y la electrónica son las principales contribuyentes a que las bases de datos tengan formato electrónico, el cual ofrece muchas soluciones al problema de almacenar datos. Para almacenar y después acceder a los datos de forma rápida, existen los Sistemas Gestores de Bases de Datos.

2.4.1. Tipos de Bases de Datos

Las bases de datos pueden clasificarse de acuerdo al criterio elegido, en este caso, según:

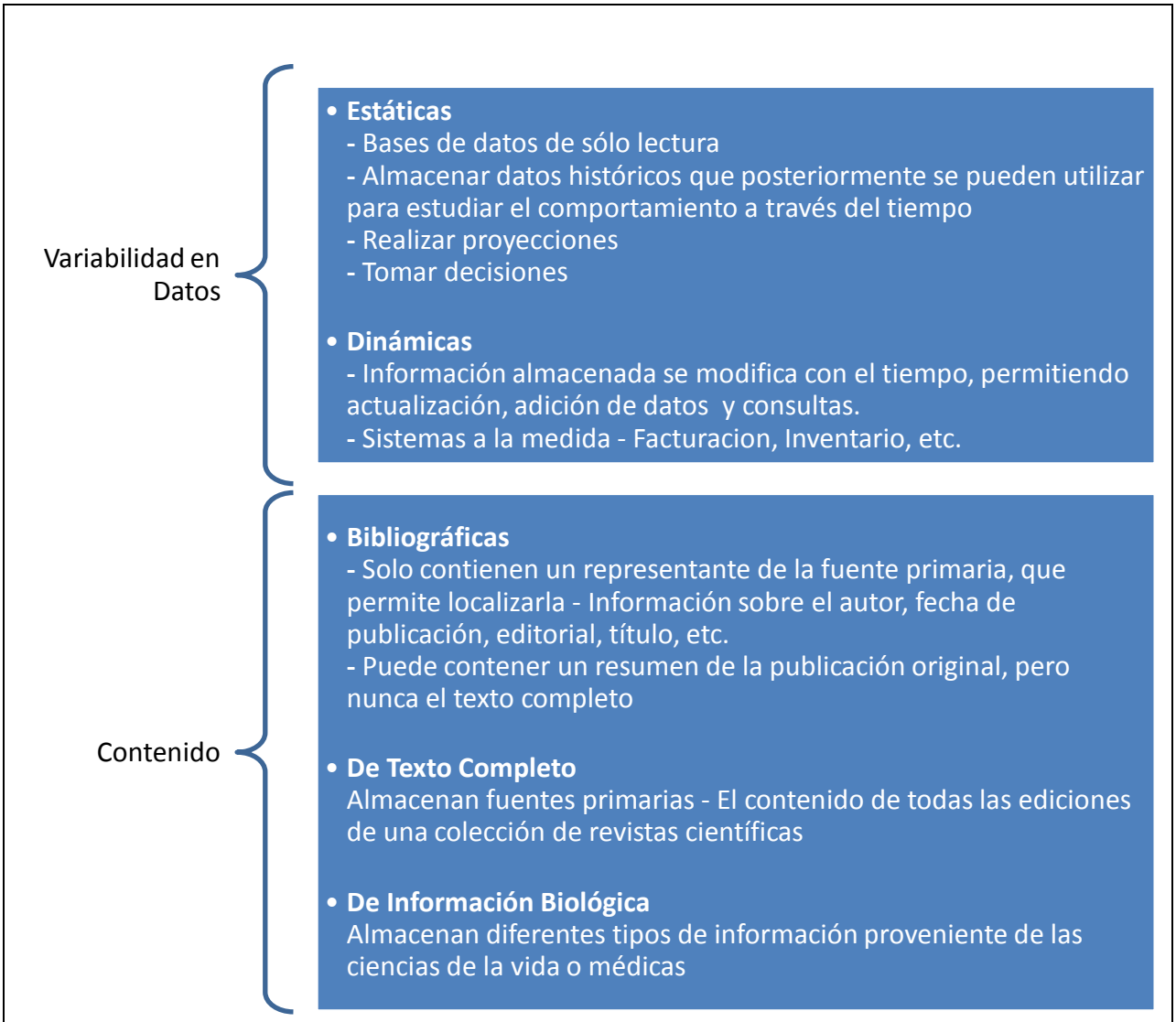


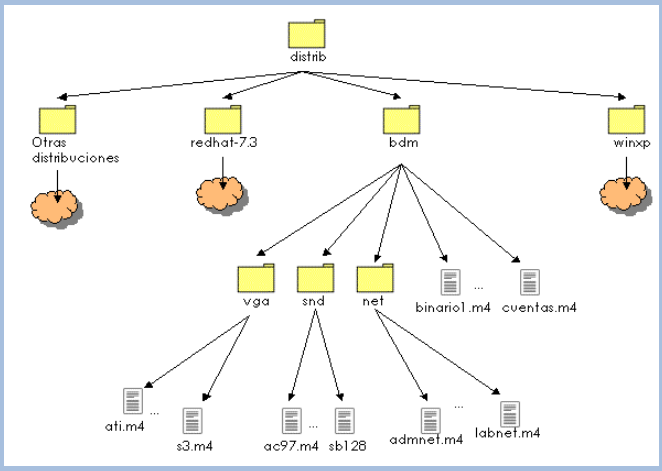
Figura 12: Tipos de Bases de Datos

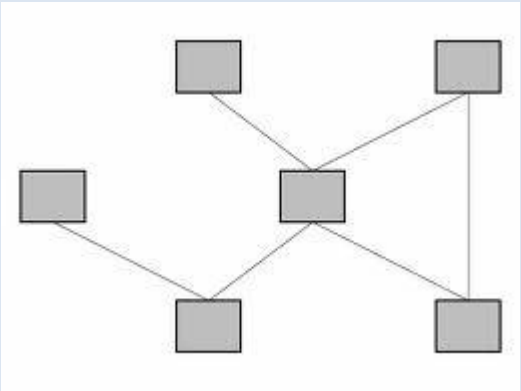
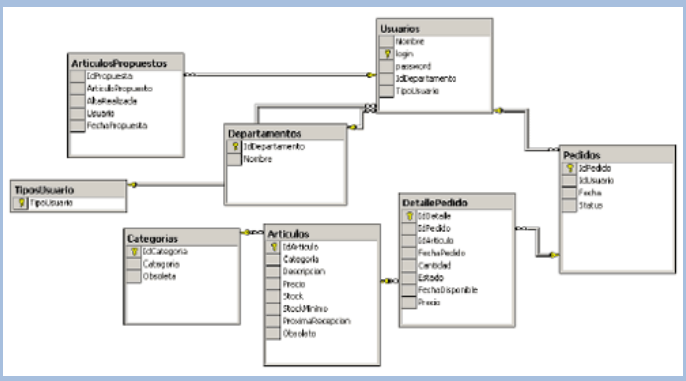
Fuente: http://www.basesdedatos.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

2.4.2. Modelos de Bases de Datos

Aparte de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos. Un modelo de datos es una descripción de un contenedor de datos (donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores.

Modelos	
JERÁRQUICAS	• Almacenan su información en una estructura jerárquica. Los datos se organizan en una forma en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.  • Útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.
DE RED	Muy parecidas a las Jerárquicas, con ciertas diferencias:

	• Se permite que un mismo nodo tenga varios padres.  • Modelo utilizado en su mayoría por programadores más que por usuarios finales.
RELACIONAL	• Modelar problemas reales y administrar datos dinámicamente. • Su idea fundamental es el uso de relaciones.  • Cada relación como si fuese una tabla que está compuesta por registros (filas), que representarían las tuplas, y campos (columnas).
ORIENTADAS A OBJETOS	Almacena en la base de datos los objetos completos (estado y comportamiento). Incorpora todos los conceptos importantes del paradigma de objetos: • **Encapsulación** – Ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos. • **Herencia** – Heredar comportamiento dentro de una jerarquía de

	clases. • **Polimorfismo** – Aplicar una operación a distintos tipos de objetos. Los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos.
DOCUMENTALES	Permiten la indexación a texto completo, y realizar búsquedas más potentes.
DEDUCTIVAS	Permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos.

Tabla 19: Modelos de Bases de Datos

Fuente: http://www.basesdedatos.org/

Elaborado Por: Ricardo Armas / Daniela Larrea

2.4.3. Sistema Gestor de Base de Datos

"Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos." (http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html, Sara Alvarez, Septiembre 2011).

Ayuda a realizar las siguientes acciones:

- Definición de los datos
- Mantenimiento de la integridad de los datos dentro de la base de datos
- Control de la seguridad y privacidad de los datos
- Manipulación de los datos

Los gestores de base de datos son un conjunto de programas no visibles al usuario final que se encargan de la privacidad, la integridad, la seguridad de los

datos y la interacción con el sistema operativo. Proporcionan una interfaz entre los datos, los programas que los manejan y los usuarios finales.

Cualquier operación que el usuario hace contra la base de datos está controlada por el gestor. El gestor almacena una descripción de datos en un Diccionario de Datos, así como los usuarios permitidos y los permisos asignados a cada uno.

Siempre tiene que haber un usuario administrador encargado de centralizar todas las tareas que se realizan sobre la base de datos.

2.4.4. Administrador de la Base de Datos

Es una persona o grupo de personas responsables del control del sistema gestor de base de datos. Las principales tareas de un administrador son:

- Definición del esquema lógico y físico de la base de datos
- Definición de las vistas de usuario
- Asignación y edición de permisos para los usuarios
- Mantenimiento y seguimiento de la seguridad en la base de datos
- Mantenimiento general del sistema gestor de base de datos

2.4.5. Microsoft SQL-Server

SQL-Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL.

Versión	Año	Nombre de la versión	Nombre clave
1.0 (OS/2)	1989	SQL Server 1-0	SQL
4.21 (WinNT)	1993	SQL Server 4.21	SEQUEL

6.0	1995	SQL Server 6.0	SQL95
6.5	1996	SQL Server 6.5	Hydra
7.0	1998	SQL Server 7.0	Sphinx
-	1999	SQL Server 7.0 OLAP Tools	Plato
8.0	2000	SQL Server 2000	Shiloh
8.0	2003	SQL Server 2000 64-bit Edition	Liberty
9.0	2005	SQL Server 2005	Yukon
10.0	2008	SQL Server 2008	Katmai
10.5	2010	SQL Server 2008 R2	Kilimanjaro

SQL-Server 2005

Esta solución de administración y análisis de datos ofrece seguridad, escalabilidad y disponibilidad mayores a las aplicaciones de datos empresariales y analíticas, a la vez que las hace más fáciles de crear, desplegar y administrar. Ofrece una solución integrada de administración y análisis de datos que ayuda a las organizaciones de cualquier magnitud a realizar lo siguiente:

- Crear, desplegar y administrar aplicaciones empresariales más seguras, escalables y confiables.
- Maximizar la productividad de TI mediante la reducción de la complejidad y el soporte de aplicaciones de bases de datos.
- Compartir datos en múltiples plataformas, aplicaciones y dispositivos para facilitar la conexión de sistemas internos y externos.
- Controlar los costos sin sacrificar el rendimiento, la disponibilidad, la escalabilidad o la seguridad.

Potencia su infraestructura de datos en tres áreas clave: administración de datos empresariales, productividad del encargado del desarrollo e inteligencia empresarial.

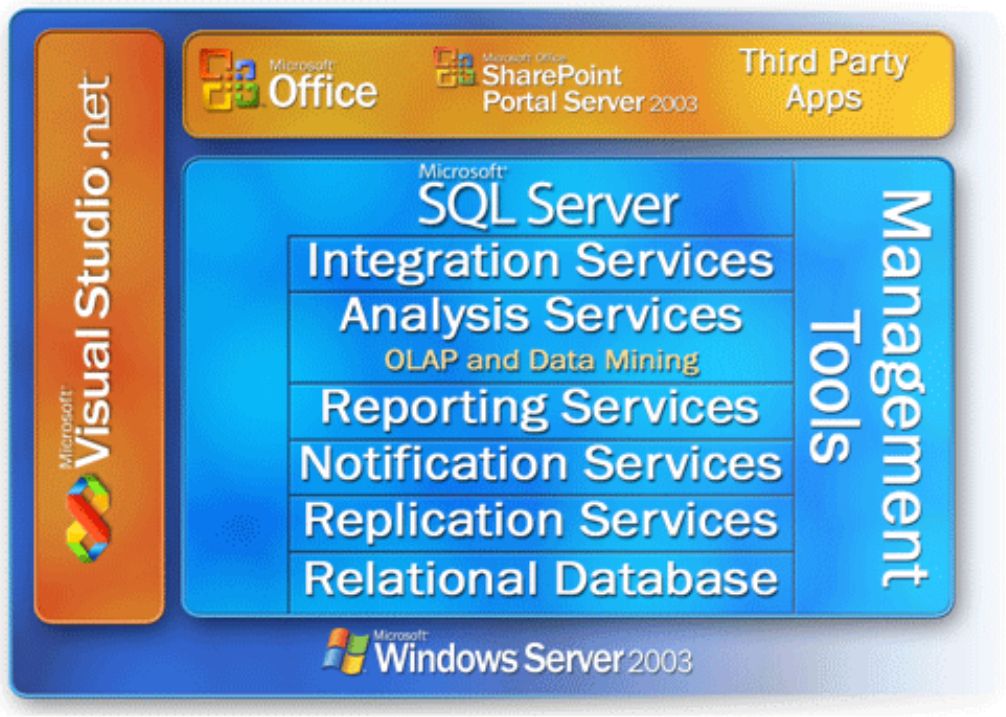


Figura 13: Componentes básicos de SQL Server 2005

Fuente: http://www.microsoft.com/spain/sql/productinfo/overview/what-is-sql-server.mspx

Elaborado Por: ©2011 Microsoft Corporation. Todos los derechos reservados.

El diagrama de la Figura 13, ilustra los componentes básicos en SQL Server 2005, muestra cómo es una parte importante de Windows Server y se integra con la plataforma Microsoft Windows, incluidos Microsoft Office y Visual Studio, para ofrecer soluciones que aportan datos a la organización.

2.5. Selección para uso en Caso Práctico

Hemos escogido utilizar la metodología de Extreme Programming con la técnica Pair Programming por los beneficios que brinda tanto para el usuario final como para nosotros que somos el equipo de desarrollo y también porque sentimos la necesidad de profundizar más sobre esta forma de trabajo.

A continuación en la Tabla 13, se presentan los beneficios que la metodología brindará a los involucrados del proyecto:

SELECCIÓN DE LA METODOLOGÍA	
Desarrolladores	Facilidad para pedir aclaraciones de los requerimientos en cualquier momento, ya que el usuario es miembro activo del equipo.
Usuario Final	Está siempre al tanto del progreso del proyecto y el estado real del software.
Usuario Final	Contar con un producto completamente funcional cada semana.
Todos los involucrados	Se cuenta con una retroalimentación frecuente entre todos los miembros del equipo.
Todos los involucrados	Comunicación abierta durante todo el proceso de desarrollo. Reuniones frecuentes para intercambio de ideas.

Desarrolladores	El diseño y la codificación siguen un concepto de simplicidad en todo momento.
Usuario Final	Los requerimientos pueden ser cambiados o modificados en cualquier momento.
Desarrolladores	Facilidad para realizar la estimación del esfuerzo y recursos para la implementación y desarrollo.
Desarrolladores	Su esfuerzo se centra solo en lo requerimientos planeados para la entrega por lo tanto se preocupan más por la calidad y simplicidad del código.
Usuario Final	Libertad para decidir si se pone o no en producción las versiones que son entregadas por el equipo de desarrollo.
Usuario Final	El usuario según su conocimiento y experiencia provee de casos de prueba para garantizar que los procesos desarrollados cumplan con dichas pruebas.
Todos los involucrados	No es necesario que se sigan todas las prácticas de la metodología, se puede utilizar las que más se ajusten al proyecto siempre teniendo en cuenta que al eliminar pasos o practicas se puede ir más rápido pero tal vez no se obtengan los mejores resultados.

Tabla 13: Selección de Metodología

Fuente: Propia

Elaborado Por: Ricardo Armas / Daniela Larrea

A continuación en la Tabla 14, se presentan los beneficios que la técnica brindará a los involucrados del proyecto:

SELECCIÓN DE LA TÉCNICA	
Desarrolladores	Intercambio de ideas y habilidades que incrementarán el conocimiento.
Desarrolladores	Tienen dominio general sobre todos los procesos del software, cualquiera puede modificar cualquier parte del código.
Desarrolladores	El intercambio de roles beneficia en gran medida a la forma de trabajo y mejora la disciplina.
Todos los involucrados	Se completan tareas más rápidamente al evitar totalmente distracciones y posibles retrasos por desconocimiento o falta de ideas.
Todos los involucrados	El trabajo en parejas es más productivo que el individual porque evita grandes errores de concepto.
Todos los involucrados	La calidad del software desarrollado usando esta técnica es mayor.

Tabla 14: Selección de la Técnica

Fuente: Propia

Elaborado Por: Ricardo Armas / Daniela Larrea

Para el presente proyecto se ha dispuesto que se le entregue al cliente una nueva versión del software cada semana, la cual será puesta en producción según el criterio del jefe de proyecto y según el avance que tengan las nuevas versiones, para que se realicen pruebas del sistema y se comunique a los desarrolladores sobre posibles nuevos requerimientos o modificaciones que se requieran. Las modificaciones a procesos ya desarrollados tendrán mayor prioridad que los nuevos requerimientos aunque podrían existir excepciones.

Hemos decidido trabajar con el lenguaje de programación orientado a objetos Java por el conocimiento que se tiene en el mismo, los beneficios que presenta tanto para el proyecto, los usuarios y los desarrolladores, y la compatibilidad que tiene con la herramienta de generación de reportes para el monitoreo.

A continuación en la Tabla 21, se presentan los beneficios que ofrece el lenguaje escogido para el desarrollo.

SELECCIÓN DEL LENGUAJE - BENEFICIOS	
Desarrolladores	• Conocimiento y experiencia previos disminuyen la dificultad y tiempos de entrega. • Compatibilidad al 100% con iReport (herramienta de generación de reportes para monitoreo desarrollada en Java), permite que las llamadas y comunicación entre el aplicativo y la herramienta no tenga conflictos. • Al ser un lenguaje orientado a objetos, el mantenimiento se lo hace a un nivel general, es decir, cualquier cambio solicitado, se lo hace una sola vez y automáticamente se afectan los demás participantes del proceso.
Usuario Final	• Al ser un lenguaje OpenSource, el costo que se debe asumir por la

	utilización del mismo para desarrollar cualquier aplicativo es nulo. • Al ser un lenguaje multiplataforma, no importa el sistema operativo de la(s) máquina(s) en la(s) que se va a instalar, ni se tiene q realizar ninguna adecuación en la(s) misma(s). • Mantenimientos rápidos por lo explicado anteriormente. • Como es un lenguaje orientado principalmente a aplicaciones Web, es muy fuerte en temas de seguridad y encriptación, por lo tanto el manejo de usuarios y accesos es restringido y controlado
Proyecto	• Mantenimientos rápidos. • Mayor seguridad. • Mejores tiempos de respuesta.

Tabla 21: Selección del Lenguaje – Beneficios

Fuente:

Elaborado Por: Ricardo Armas / Daniela Larrea

Actualmente, en la sede del INIAP, se cuenta con un sistema electrónico que registra en una base de datos las medidas periódicas que se realizan para la Temperatura y Humedad en los diferentes cuartos. La base de datos que utiliza el sistema electrónico es una Microsoft SQL-Server 2005.

El sistema de monitoreo a desarrollar utilizará los datos almacenados en la base para generar los reportes, gráficos, y estadísticas requeridos por el usuario. Por esta razón la base que se utilizará para el sistema a desarrollar es Microsoft SQL-Server 2005.

CAPITULO III

CASO DE ESTUDIO

3. CASO DE ESTUDIO

En este capítulo se describirá las principales características del Instituto Nacional Autónomo de Investigaciones Agropecuarias (INIAP), posteriormente se detallará el proceso que se realiza en la sede del INIAP en Santo Domingo para la extracción de semilla de palma africana y por último se especifican los equipos utilizados en el proceso, su ubicación y los parámetros medidos por cada uno de estos.

3.1 Presentación de la Empresa

Instituto Nacional Autónomo de Investigaciones Agropecuarias



Ubicación:

Km. 38 vía Santo Domingo.
Quinindé, cantón La Concordia,
provincia de Esmeraldas.



Actividad:

Generar y proporcionar tecnologías apropiadas, productos, servicios y capacitación especializada para contribuir al desarrollo sostenible de los sectores agropecuario, agroforestal y agroindustrial.

Servicios:

Venta de plantas y semillas de palma africana; y capacitación a través del núcleo de Transferencia de Tecnología.



El INIAP tiene una estructura orgánica por procesos, adaptada a las condiciones especificas de su actividad técnica científica, para lo cual ha establecido los siguientes procesos:

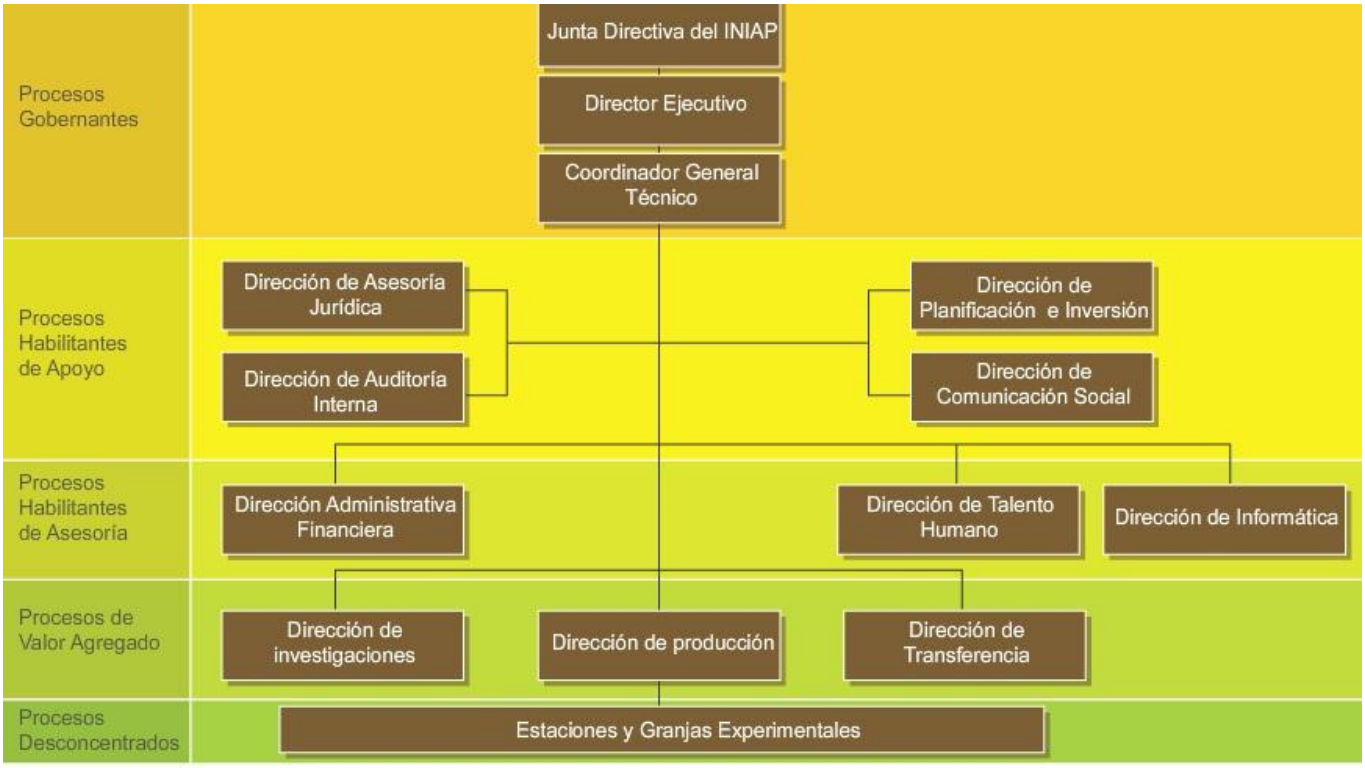


Tabla 22: Presentación del Insituto INIAP

Fuente: http://www.iniap.gob.ec/

Elaborado por: Ricardo Armas / Daniela Larrea

Características de la Empresa:

- El Instituto mantiene un serio compromiso con la seguridad alimentaria de la población y con la necesidad de lograr una producción agropecuaria competitiva, para afrontar con éxito los desafíos de la globalización, entre los cuales están las exigencias de los mercados internacionales que demandan productos agropecuarios de alta calidad, que solamente se logran con la aplicación de tecnologías generadas y adaptadas a las condiciones agroecológicas del país.
- En la sede Santo Domingo, se realizan investigaciones de: Palma africana, caucho, maracuyá y cacao fino de aroma.

3.2 Recolección de Datos

3.2.1 Justificación

El motivo principal por el cual se realizó esta aplicación es la necesidad que tienen los investigadores y personal en general que labora en esta sede específicamente en el proyecto de investigación de Palma Africana, para poder generar reportes de cómo se está llevando a cabo el monitoreo de las distintas etapas que constituyen el proyecto de conservación, las etapas en las es necesario un monitoreo son:

- Almacenamiento.
- Germinación.
- Isotérmico.

Para lo cual cuentan con nueve cuartos con equipos de monitoreo de: Temperatura y Humedad, según sea necesario. Estos equipos están controlados por un sistema electrónico el cual envía los registros que se capturan a una base de datos cada determinado tiempo, generalmente cada minuto. En la sección 3.2.4 se indica a detalle su funcionamiento. De aquí en adelante a estos elementos se los llamará simplemente "cuartos"

Con la implementación del sistema de monitoreo, se conseguirá llevar un registro ordenado e histórico del cambio que se dio en los diferentes cuartos, para saber si están o no funcionando correctamente los equipos y mejorar en gran medida los resultados de la investigación lo cual ayudará a cumplir con el objetivo de conservar esta planta.

El sistema brindará también la posibilidad de obtener reportes sobre:

- Los usuarios del sistema.
- Las alarmas que se dieron en determinado tiempo y determinado cuarto.
- Cuándo se dio calibración a los equipos o
- Cuándo es necesario dar mantenimiento a los mismos.

Los reportes son necesarios para todos los empleados, para los que trabajan directamente en los cuartos que son técnicos e Ingeniero Agrónomos, que les servirá para mantener registros históricos del proceso que realizan con las semillas de la palma africana.

3.2 Antecedentes

Actualmente, en la sede, se cuenta con un sistema electrónico que mediante la conexión a los equipos registra en una base de datos Microsoft SQL SERVER 2005 las medidas periódicas que se realizan para la Temperatura y Humedad en los diferentes cuartos, de la misma manera registran:

- Las alarmas que se dieron cuando ocurrió una subida o bajada fuera del nivel permitido para estas dos medidas.
- Por cuanto tiempo están prendidos los equipos.
- La fecha y hora exacta en que fueron apagados o prendidos.
- Cuando se dieron sobrecargas en los equipos.
- Cambios de Presión en los cuartos.

Todos estos registros se almacenan en una base de datos pero el sistema electrónico no permite visualizar los históricos en forma gráfica o en tablas, sino simplemente se ve el monitoreo en vivo de los cuartos.

3.2.1 Proceso de Producción de semilla de Palma Africana

Para entender de mejor manera el proceso que se realiza con la palma africana en esta sede es necesario conocer más sobre este tipo de planta:

Palma Africana

Es una planta de origen Africano, específicamente del golfo de Guinea, es un cultivo de clima cálido y a pesar de que su vida productiva es de 50 años o más, sus primeros frutos se obtienen pasados aproximadamente 2 años y medio de su siembra. En sus años de mayor producción (20 – 30 años) miden 12 metros de altura pero después dejan de ser rentables por la altura de más de 40 metros que alcanzan. En la Figura 14 se observa una plantación de palma africana.



Figura 14: Palma Africana

Fuente: http://pepascientificas.blogspot.com/2010/02/12-preguntas-y-respuestas-sobre-la.html

Se caracterizan por el aceite que se extrae de la pulpa de su fruto, el cual es uno de los aceites más consumidos en el mundo con aplicaciones en productos comestibles, no comestibles, jabones, cosméticos, entre otros.

En la Figura 15 se observan las partes que conforman el fruto de la palma africana y en la Tabla 23 se describe a cada una.



Figura 15: Partes del Fruto de Palma Africana

Fuente: http://www.infoagro.com/herbaceos/oleaginosas/palma_africana_aceitera_coroto_de_guinea_aabora.htm

PARTES DEL FRUTO DE LA PALMA AFRICANA	
1. Cáscara	Es el exocarpio, es decir la capa más externa que recubre al fruto.
2. Cuesco	Es el endocarpio, la capa de consistencia leñosa que rodea a la semilla.
3. Pulpa	Es el mesocarpio, la parte carnosa que contiene el aceite y es de tejido fibroso.
4. Embrión	La parte del fruto que dará lugar a una nueva planta.
5. Almendra	Es el endospermo, un depósito de alimento aceitoso para el embrión.

Tabla 23: Partes del fruto de la Palma Africana
Fuente: http://palma.aceitescomestibles.com/index.php?option=com_content&view=article&id=50&Itemid=57
Elaborado por: Ricardo Armas / Daniela Larrea

Los frutos de palma africana crecen en racimos como se observa en la Figura 16:



Figura 16: Racimo de frutos de Palma Africana

Fuente: http://www.infoagro.com/herbaceos/oleaginosas/palma_africana_aceitera_coroto_de_guinea_aabora.htm

Conceptos Importantes

A continuación en la Tabla 24 se presentan algunos conceptos importantes a conocer para el correcto entendimiento del proceso de obtención de la semilla de palma africana.

Radícula	Parte del embrión que emerge primero y al desarrollarse se constituye en la raíz.
Plúmula	Es la parte que corresponde al tallo pero cuando aún no alcanza su forma y estructura definitiva.
Germinación	Es el proceso donde bajo ciertas condiciones favorables, la semilla recupera la capacidad biológica de crecimiento embrionario.
Temperatura	Magnitud escalar relacionada con la actividad molecular de la materia. A mayor movimiento de moléculas mayor temperatura y viceversa.

Humedad Relativa	Cantidad de vapor de agua en el aire, relación entre la presión parcial de vapor actual y la presión de vapor de saturación. Está expresada en porcentaje (%).

Tabla 24: Conceptos Importantes en el Proceso de Producción de semilla de Palma Africana

Fuente: www.lamolina.edu.pe/agronomia/dhorticultura/html/.../lucia.doc

Elaborado por: Ricardo Armas / Daniela Larrea.

Proceso de Producción

1. **Cosecha de Racimos:** proceso necesario para la obtención de las semillas que serán usadas posteriormente en el proceso de germinación cuando lleguen a su estado normal de maduración.
2. **Desgrane:** proceso de extracción manual de los frutos.
3. **Fermentación:** los frutos son colocados en bolsas de plástico las cuales son sumergidas en depósitos de agua para facilitar el desprendimiento de la cascara.
4. **Despulpado:** proceso en el cual se extraen la cáscara y la pulpa utilizando generalmente procesos mecánicos debido a la gran cantidad de semillas. Los despulpadores cuentan con mallas metálicas que giran a una velocidad determinada.
5. **Secamiento:** necesario para contar con una humedad adecuada en la semilla antes de ser almacenada, no pasa de los 2 días.
6. **Selección:** solo las semillas que alcanzaron un tamaño adecuado son elegidas porque las muy pequeñas a menudo no germinan.
7. **Almacenamiento:** las semillas seleccionadas son almacenadas en bolsas de polietileno selladas herméticamente y son colocadas en estantes al interior de un cuarto donde la temperatura este entre 18 °C a 20 °C y con una humedad no mayor a 60%.
8. **Calentamiento:** después de tener las semillas remojadas por 8 días, se procede a secarlas por no más de 8 horas a la sombra para evitar el secamiento excesivo, posteriormente se coloca nuevamente las semillas en bolsas de polietileno dejando un espacio de aire considerable y sellándolas

con bandas plásticas las bolsas son colocadas en el interior de un cuarto con temperatura de hasta 39°C y condiciones óptimas de humedad, por un período de 80 días.

Pasados este período se vuelve a realizar un remojo y secado en la sombra pero no más de 4 horas.

9. **Germinación:** en esta etapa se ponen nuevamente las semillas dentro de bolsas de polietileno selladas pero dejando un espacio libre de aire. En el cuarto de germinación se pone las bolsas a temperatura ambiente por 30 días. En los primeros 10 días se puede ver una expansión de la radícula en forma de pequeños puntos blancos sobre la semilla. Todas las semillas deben germinar por lo tanto son dejadas en las bolsas hasta que se cumpla el proceso.

10. **Selección de semilla germinada:** las semillas que terminan el proceso de germinación son las que tienen visible la plúmula y radícula. Estas semillas son vendidas directamente a los palmicultores o se siembran en viveros en la misma sede de Santo Domingo para vender directamente las plantas posteriormente. En la Figura 17 se observa el proceso con imágenes de cada una de las etapas:



Figura 17: Proceso de extracción de la semilla de Palma Africana

Fuente: http://redepalma.blogspot.com/2009/03/fruto-de-la-palma-africana-el-fruto-de.html

Elaborado por: Ricardo Armas / Daniela Larrea

3.2.2 Equipos y Parámetros en los cuartos

Después de entender el proceso de producción y extracción de semillas de palma africana es necesario describir como están equipados los cuartos que son utilizados para cada una de las etapas de producción.

Existen cuartos de:

- Almacenamiento
- Germinación
- Isotérmicos

Donde en cada uno existen diferentes equipos según los parámetros necesarios a considerar. Primero se explican en general los equipos instalados en los cuartos para posteriormente realizar una descripción de que equipos se encuentran en cada cuarto.

Descripción de Equipos

En la Tabla 25 se detalla a los equipos necesarios para la captura de parámetros:

Unidad Evaporadora 	Equipo necesario para absorber calor y enfriar el cuarto mediante la vaporización de líquidos refrigerantes.
Conformada por: • Compresor: comprime el vapor recibido de la evaporadora y lo	**Unidad Condensadora**

descarga a una alta presión y temperatura. • Condensador: intercambiador y disipador de calor del refrigerante al aire exterior.	
Unidad Manejadora de Aire 	Se encarga de regular la temperatura mediante un ventilador y un motor.
Calienta el aire elevando la temperatura.	**Resistencias Calefactoras** 

Tabla 25: Equipos necesarios para los cuartos.

Fuente: http://www.vendasa.com/faqs.swf - http://www.stego.de/es/productos/resistencia-calefactora.html

Elaborado por: Ricardo Armas / Daniela Larrea.

La importancia de estos equipos es muy alta puesto que se asegura contar con las condiciones necesarias para que se lleve a cabo el proceso de extracción de las semillas de la mejor manera. A continuación, en la Tabla 26 se detalla cómo están distribuidos los equipos en los diferentes cuartos:

CUARTO	EQUIPOS
Almacenamiento 1	Unidad Condensadora N°1 Unidad Evaporadora N°1
Almacenamiento 2	Unidad Condensadora N°2 Unidad Evaporadora N°2
Isotérmico 1	Unidad Manejadora de Aire N°1 Resistencia Calefactora N°1 - 2
Isotérmico 2	Unidad Manejadora de Aire N°2 Resistencia Calefactora N°3 – 4
Isotérmico 3	Unidad Manejadora de Aire N°3 Resistencia Calefactora N°5 – 6
Isotérmico 4	Unidad Manejadora de Aire N°4 Resistencia Calefactora N°7 - 8
Germinador 1	Unidad Condensadora N°3 Unidad Evaporadora N°3 Resistencia Calefactora N°9
Germinador 2	Unidad Condensadora N°4 Unidad Evaporadora N°4 Resistencia Calefactora N°10
Germinador 3	Unidad Condensadora N°5

	Unidad Evaporadora N°5 Resistencia Calefactora N°11

Tabla 26: Distribución de equipos en cuartos

Fuente: INIAP

Elaborado por: Ricardo Armas / Daniela Larrea

Condiciones de Fallo

Los equipos que se encuentran distribuidos en los 9 cuartos necesitan de un mantenimiento y monitoreo constante, pero pueden ocurrir fallas por factores climáticos extremos o por fallas de los equipos. Los parámetros de temperatura y humedad deben estar dentro de un rango prestablecido en base a investigaciones realizadas sobre el proceso de extracción de semillas de palma africana, en base a la información entregada para el desarrollo de este proyecto se determina que pueden existir fallas por:

- Temperatura alta.
- Temperatura baja.
- Humedad Alta.
- Humedad baja.

En cuanto a la presión también pueden ocurrir fallas por posibles obstrucciones lo que impide una buena circulación de refrigerante, por lo tanto puede existir:

- Presión alta.
- Presión baja.

Los equipos cuentan con protecciones frente a sobrecargas, cuando es necesario que estas actúen de igual manera se producen alarmas en los equipos para:

- Sobrecarga de la unidad condensadora.
- Sobrecarga de la unidad Evaporadora.
- Sobrecarga en la unidad Manejadora de Aire.

3.2.3 Sistema Electrónico SCADA

"Los sistemas SCADA (Supervisory Control And Data Adquisition) son aplicaciones de software, diseñadas con la finalidad de controlar y supervisar procesos a distancia. Se basan en la adquisición de datos de los procesos remotos." *(Ing. Henry Mendiburu Díaz, Sistemas SCADA, http://hamd.galeon.com).*

SCADA es una aplicación de software, diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos y controlando el proceso de forma automática desde una computadora. Aparte de esto, envía la información generada en el proceso de producción a usuarios tanto del mismo nivel como hacia otros supervisores dentro de la empresa, es decir, permite la participación de otras áreas como por ejemplo: control de calidad, supervisión, mantenimiento, etc.

A continuación en la Figura 18 se puede observar la centralización que se puede controlar y supervisar a través de SCADA.

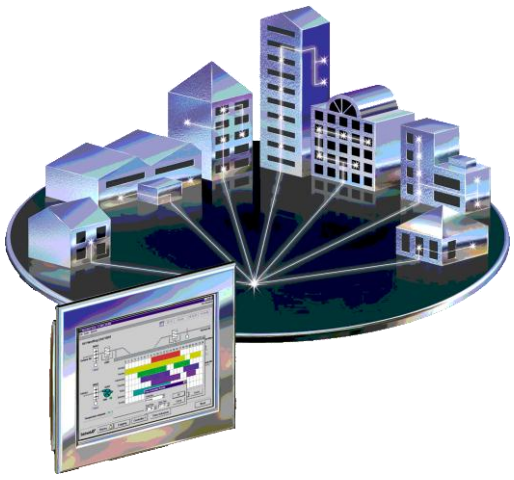


Figura 18: Sistema SCADA Edificio Inteligente

Fuente: http://control-accesos.es/scada/%C2%BFque-es-un-sistema-scada

Elaborado Por: © 2011 Blog de control de accesos. Derechos reservados.

Ítems principales de SCADA

Cada uno de los ítems de SCADA involucra muchos subsistemas, por ejemplo:

- **Adquisición de datos**

  La adquisición de los datos puede estar a cargo de un PLC (Controlador Lógico Programable) el cual toma las señales y las envía a las estaciones remotas. Otra forma podría ser que una computadora realice la adquisición de los datos vía hardware especializado y luego esa información la transmita hacia un equipo de red a través de su tarjeta de red.

- **Supervisión y Control**

  Las tareas de Supervisión y Control generalmente están más relacionadas con el software SCADA, en el cual el operador puede visualizar en la pantalla del computador de cada una de las estaciones remotas que conforman el sistema, los estados de ésta, las situaciones de alarma y tomar acciones físicas sobre algún equipo lejano.

  Todo esto se ejecuta normalmente en tiempo real, y están diseñados para dar al operador de planta la posibilidad de supervisar y controlar dichos procesos.

Estos sistemas actúan sobre los dispositivos instalados en la planta, como son los controladores, sensores, registradores, etc. Además permiten controlar el proceso desde una estación remota, para ello el software brinda una interfaz gráfica que muestra el comportamiento del proceso en tiempo real.

A continuación en la Tabla 27 se muestran los aspectos básicos de un sistema SCADA.

	SCADA
TIPO DE ARQUITECTURA	CENTRALIZADA
TIPO DE CONTROL PREDOMINANTE	SUPERVISORIO: Lazos de control cerrados por el operador. Adicionalmente: control secuencial y regulatorio.
TIPOS DE VARIABLES	DESACOPLADAS
ÁREA DE ACCIÓN	Áreas geográficamente distribuídas.
UNIDADES DE ADQUISICIÓN DE DATOS Y CONTROL	Remotas, PLCs.
MEDIOS DE COMUNICACIÓN	Radio, satélite, líneas telefónicas, conexión directa, LAN, WAN.
BASE DE DATOS	CENTRALIZADA

Tabla 27: Aspectos básicos de un Sistema SCADA

Fuente: http://www.kyber.cl/Energiaelectrica/scada_electrico.html

Elaborado Por: Ricardo Armas / Daniela Larrea

Generalmente se vincula el software al uso de una computadora, la acción de control es realizada por los controladores de campo, pero la comunicación del sistema con el operador es necesariamente vía computadora.

Un software SCADA debe ser capaz de ofrecer al sistema:

- Posibilidad de crear paneles de alarma, que exigen la presencia del operador para reconocer una parada o situación de alarma, con registro de incidencias.

- Generación de datos históricos de las señales de planta, que pueden ser exportados para su proceso sobre una hoja de cálculo.

- Ejecución de programas, que modifican la ley de control, o incluso anulan o modifican las tareas asociadas al proceso automatizado, bajo ciertas condiciones.

Posibilidad de programación numérica, que permite realizar cálculos aritméticos de alta complejidad.

Funciones Principales

Supervisión remota de instalaciones y equipos	Control remoto de instalaciones y equipos	Procesamiento de datos
• Permite al operador conocer el estado de desempeño de las instalaciones y los equipos alojados en la planta. • Permite dirigir las tareas de mantenimiento y estadística de fallas.	• Se puede activar o desactivar los equipos remotamente de manera automática y también manual. • **Remotamente se puede: Abrir válvulas, activar interruptores, prender motores, etc.**	• El conjunto de datos adquiridos conforman la información que alimenta el sistema. • La información adquirida es procesada, analizada, y comparada con datos anteriores, dando como resultado una información confiable y veraz.

Visualización gráfica dinámica	Generación de reportes	Representación de señales de alarma
• El sistema es capaz de brindar imágenes en movimiento que representen el comportamiento del proceso. • Pueden corresponder a curvas de las señales analizadas en el tiempo.	• El sistema permite generar informes con datos estadísticos del proceso en un tiempo determinado por el operador.	• A través de las señales de alarma se logra alertar al operador frente a una falla o la presencia de una condición fuera de lo aceptable. • Las señales de alarma pueden ser tanto visuales como sonoras.

Almacenamiento de información histórica	Programación de eventos
• Existe la opción de almacenar los datos adquiridos para analizarla posteriormente. • El tiempo de almacenamiento dependerá del operador o del administrador del programa.	• El conjunto de datos adquiridos conforman la información que alimenta el sistema. • La información adquirida es procesada, analizada, y comparada con datos anteriores, dando como resultado una información confiable y veraz.

Tabla 28: Funciones principales de SCADA

Fuente: http://hamd.galeon.com

Elaborado Por: Ricardo Armas / Daniela Larrea

Elementos de un Sistema SCADA

Todo sistema SCADA está conformado por los siguientes elementos:

- Interfaz Operador Máquinas: Es el entorno visual que brinda el sistema para que el operador se adapte al proceso desarrollado por la planta. Permite la interacción del ser humano con los medios tecnológicos implementados.

- Unidad Central (MTU): Ejecuta las acciones de mando (programadas) en base a los valores actuales de las variables medidas. La programación se realiza en lenguaje de alto nivel (como C, Basic, etc.). También se encarga del almacenamiento y procesamiento de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.

- Unidad Remota (RTU): Cualquiera de los elementos que envía algún tipo de información a la unidad central. Es parte del proceso productivo y necesariamente se encuentra ubicada en la planta.

- Sistema de Comunicaciones: Se encarga de la transferencia de información del punto donde se realizan las operaciones, hasta el punto donde se supervisa y controla el proceso. Lo conforman los transmisores, receptores y medios de comunicación.

- Transductores: Son los elementos que permiten la conversión de una señal física en una señal eléctrica y viceversa. Su calibración es muy importante para que no haya problema con la confusión de valores de los datos.

A continuación en la Figura 19 se puede ver cómo interactúan los elementos que forman parte del Sistema SCADA, con el proceso a tratar.

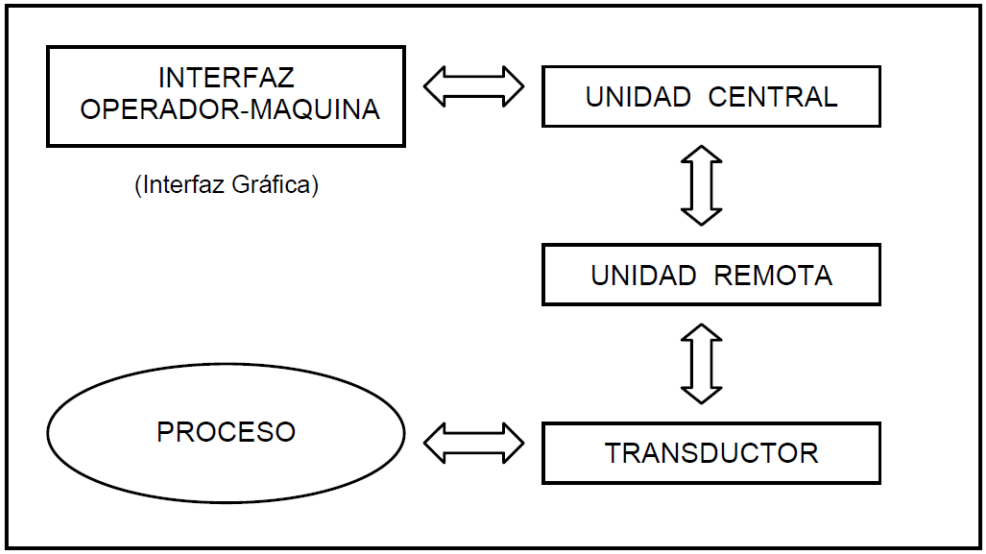


Figura 19: Esquema de los elementos de un sistema SCADA

Fuente: http://hamd.galeon.com

Elaborado Por: Henry Mendiburu Diaz

Interfaz Humano – Máquina HMI

La HMI permite a los usuarios del departamento de producción supervisar el proceso de germinación de las semillas de palma africana mediante un computador.

Para el desarrollo de la interfaz del programa se utilizó la plataforma de desarrollo WinCC y fue desarrollada por la empresa auspiciante MZ Sistemas.

Con el desarrollo e implementación de la HMI se facilitó y mejoró las tareas que los usuarios realizaban por la facilidad que ofrece la interfaz en la interacción entre el operador y el proceso. El HMI presenta pantallas de objetos con los que el usuario puede interactuar de manera amigable.

A continuación en la Figura 20 se observa la pantalla de inicio de la interfaz humano – máquina desarrollada por la empresa auspiciante. Como se puede ver, esta interfaz ayuda al operador a realizar las funciones de configuración, de control del sistema y de avisos y entrar al proceso de producción.



Figura 20: Pantalla de Inicio del HMI del Sistema SCADA

Fuente: Sistema SCADA instalado en sede del INIAP

Elaborado Por: Copia de Pantalla MZ Sistemas Eléctricos – Electrónicos.

Así mismo, dentro del sistema y de la función del Proceso de Producción, se puede ver el estado de cada uno de los cuartos de almacenamiento, isotérmico y germinación, en los cuales se presentaran las alertas cada vez que los dispositivos electrónicos detecten un alza o baja de temperatura o humedad.

A continuación en la Figura 21, se observa la pantalla del proceso de producción.



Figura 21: Pantalla de Proceso de Producción

Fuente: Sistema SCADA instalado en sede del INIAP

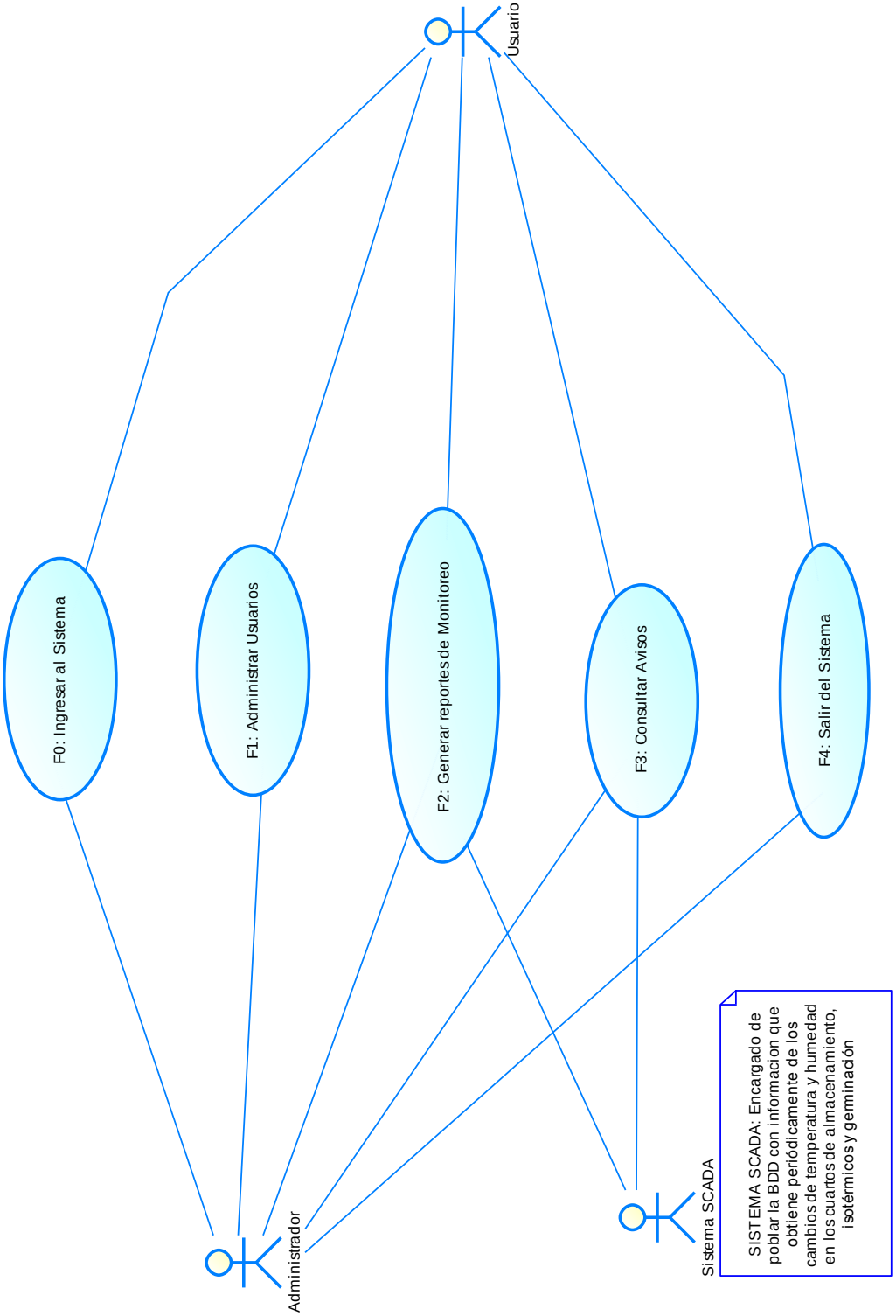
Elaborado Por: Copia de Pantalla MZ Sistemas Eléctricos – Electrónicos.

CAPITULO IV

DESARROLLO DEL PRODUCTO

4. DESARROLLO DEL PRODUCTO

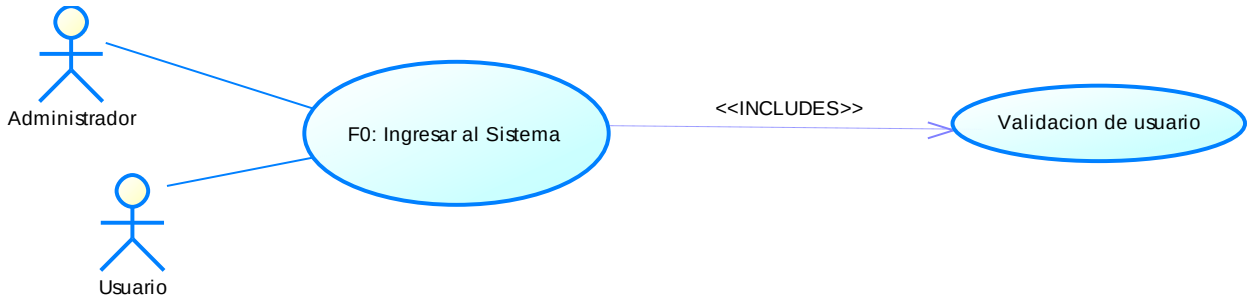
4.1 Diagramas Casos de Uso



4.2 Diagramas de Secuencias

INGRESO AL SISTEMA

F0: Ingresar al Sistema



Actores: Administrador y Usuario.

Flujo Principal:

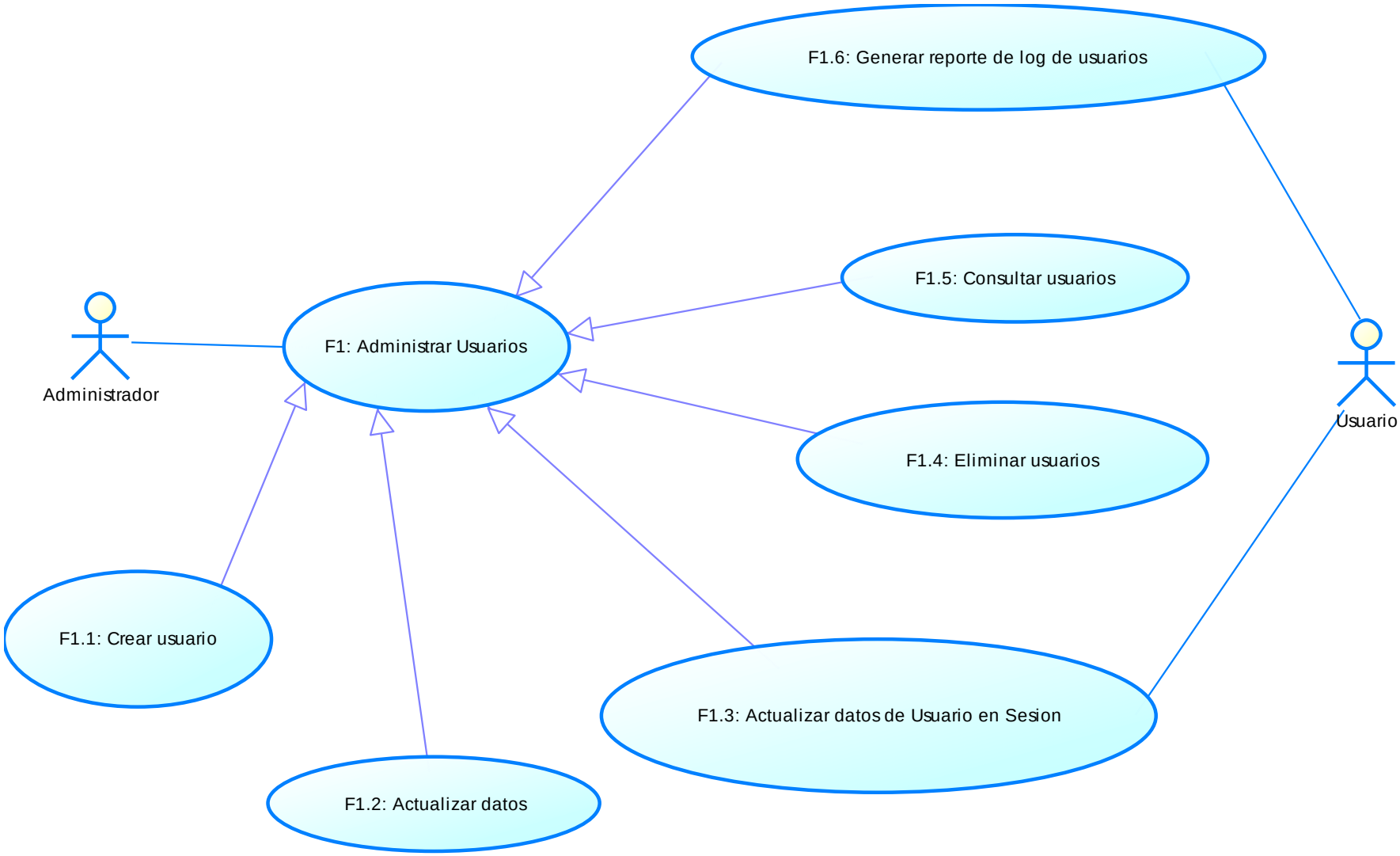
1. El actor ejecuta el programa.
2. El sistema presenta la ventana de ingreso con campos de usuario y contraseña.
3. El actor ingresa el nombre de usuario.
4. El actor pulsa botón ACEPTAR.
5. El sistema verifica existencia por medio de validación de usuario / contraseña **(E1)**
6. El sistema presenta la pantalla con el Menú Principal.

Flujo alterno:

6. Si existe en BDD y el sistema valida la información ingresada con la de la BDD → Ver Casos de Uso Administración de Usuarios (F1), Generación de Reportes de Monitoreo (F2), Consultar Avisos (F3), Salir del Sistema (F4)

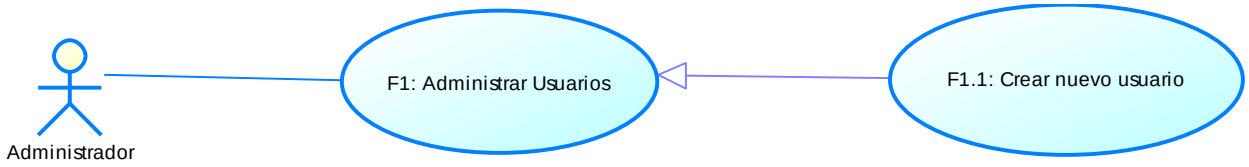
Excepciones:

- **E1:** Problemas de validación, inconsistencia de datos ingresados por el usuario contra datos almacenados en el sistema.


F1.6: Generar reporte de log de usuarios
F1.5: Consultar usuarios
F1: Administrar Usuarios
Administrador
Usuario
F1.4: Eliminar usuarios
F1.1: Crear usuario
F1.3: Actualizar datos de Usuario en Sesion
F1.2: Actualizar datos

ADMINISTRAR USUARIOS

F1.1: Crear Usuario



Flujo Principal:

1. El administrador selecciona opción de Administrar Usuarios
2. El administrador selecciona opción de Crear Usuario
3. El sistema presenta la ventana para llenar datos del nuevo usuario
4. El administrador ingresa el nombre del nuevo usuario a ingresar
5. El sistema verifica existencia **(E1)**
6. El administrador ingresa los demás datos para el nuevo usuario
7. El administrador presiona botón GUARDAR**(E2)**

Flujo alterno:

5. Si existe en BDD → Ver caso de uso Actualizar datos (F1.2) o ver caso de uso Eliminar usuarios (F1.3) o ver caso de uso Consultar usuarios (F1.4) o ver caso de uso Generar reporte de log de usuarios (F1. 5).

F1.2: Actualizar Datos

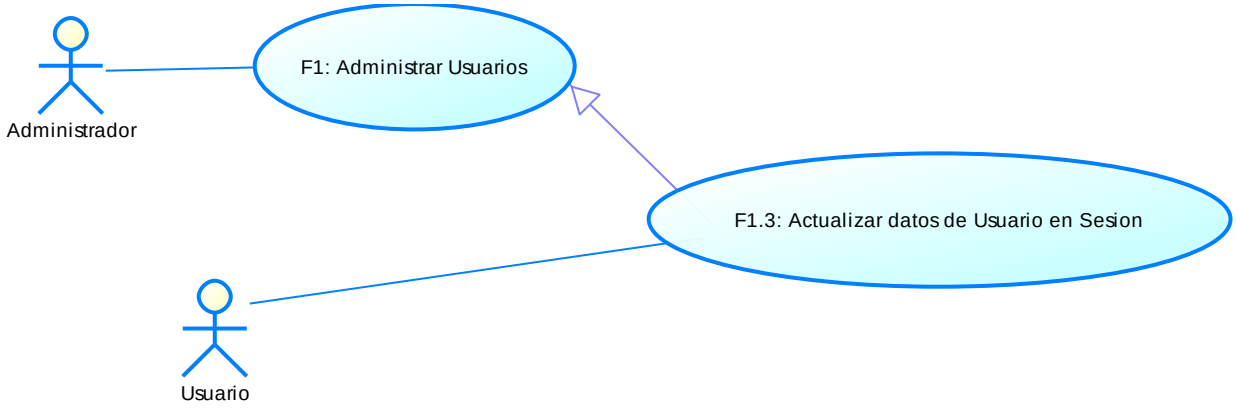


Flujo Principal:

1. El administrador selecciona opción de Administrar Usuarios
2. El administrador selecciona opción de Consultar Usuarios
3. El sistema presenta la ventana con un listado de los usuarios ingresados
4. El administrador selecciona el usuario del cual desea modificar datos
5. El administrador presiona el botón RESET CONTRASEÑA

6. El sistema despliega diálogo de confirmación de acción
7. El administrador presiona el botón SI
8. El sistema reasigna la contraseña por defecto 'abc123' al usuario en cuestión

F1.3: Actualizar Datos de Usuario en Sesión



Actores: Administrador y Usuario

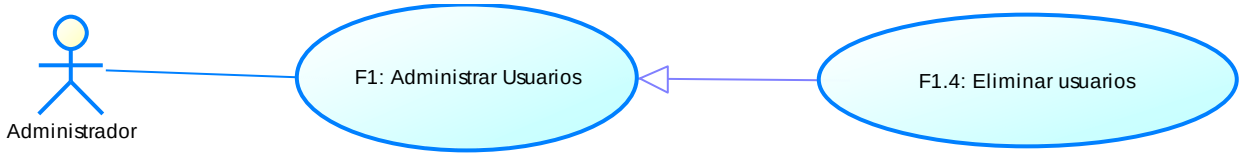
Flujo Principal:

1. El actor en sesión selecciona opción de Administrar Usuarios
2. El sistema presenta pantalla del perfil del Usuario en sesión con sus respectivos datos
3. El actor en sesión actualiza los datos de contraseña
4. El actor en sesión presiona el botón GUARDAR
5. El sistema despliega diálogo de confirmación de acción
6. El sistema valida la información de la contraseña actual almacenada en la BDD **(E1)**
7. El sistema guarda la información actualizada

Flujo Alterno:

7. Si los datos de la contraseña actual no coinciden con los almacenados en la BDD → Sistema despliega diálogo de error de contraseña actual, ver caso de uso Generar reporte de log de usuarios (F1. 5).

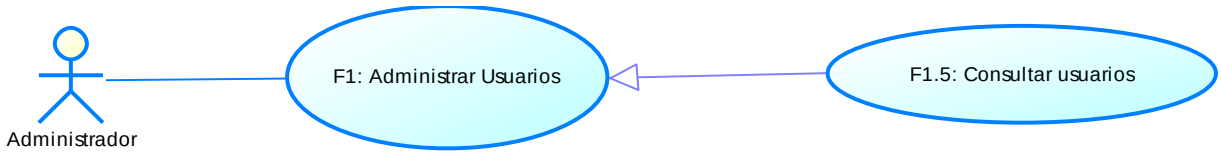
F1.4: Eliminar Usuarios



Flujo Principal:

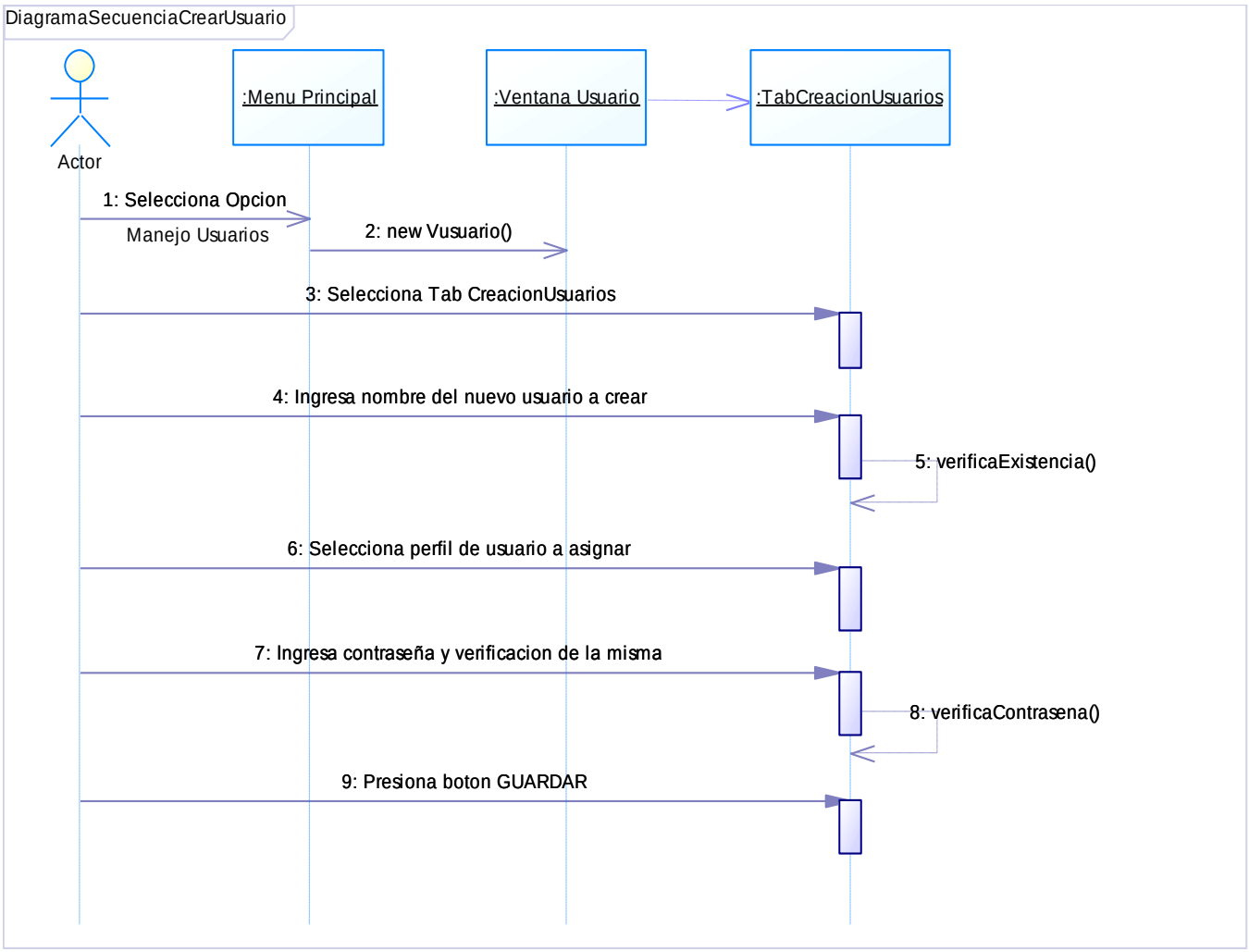
1. El administrador selecciona opción de Administrar Usuarios
2. El administrador selecciona opción de Consultar Usuarios
3. El sistema presenta la ventana con un listado de los usuarios ingresados
4. El administrador selecciona el usuario que desea eliminar
5. El administrador presiona el botón ELIMINAR
6. El sistema despliega diálogo de confirmación de acción
7. El administrador presiona el botón SI
8. El sistema elimina el usuario seleccionado de la BDD

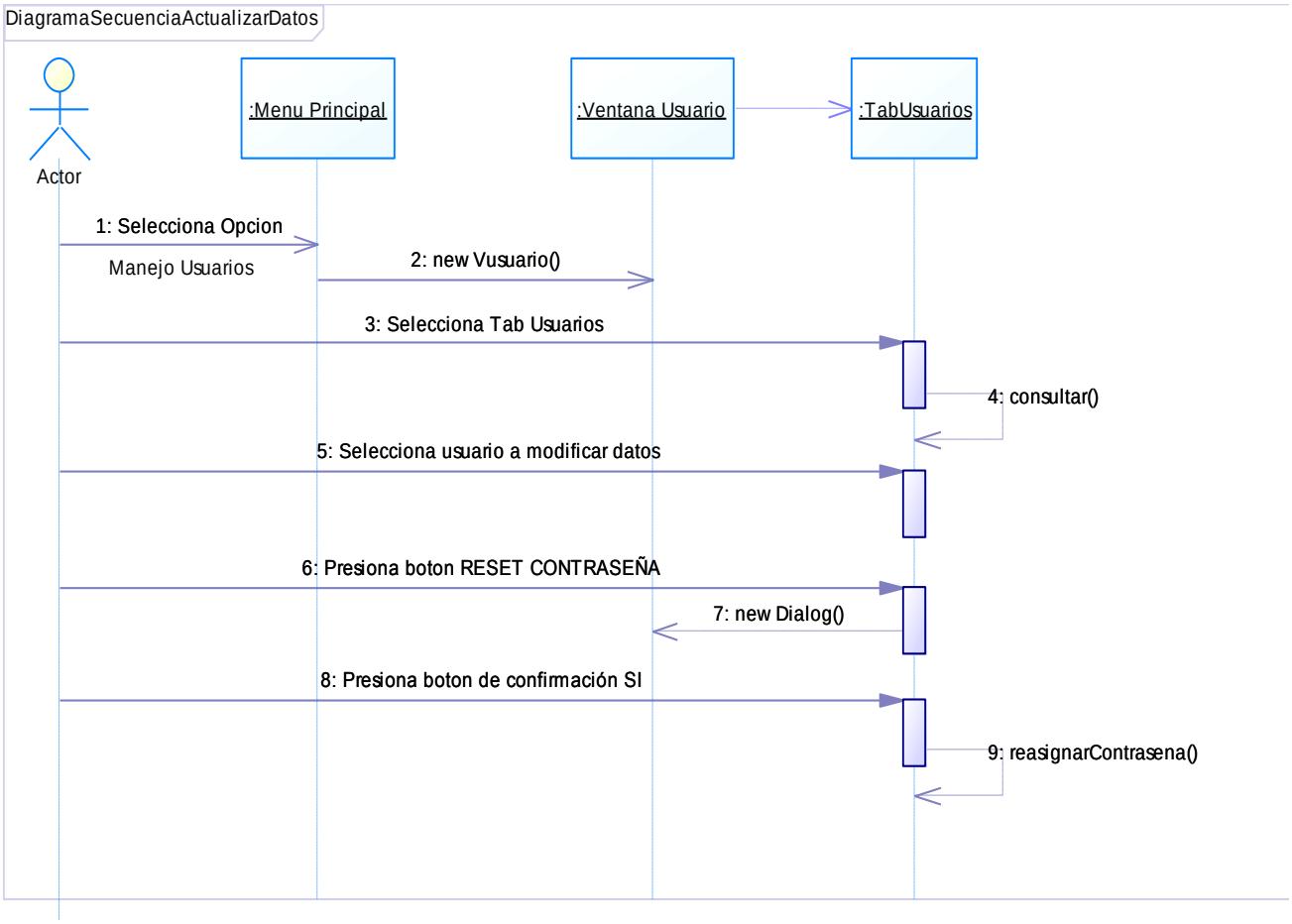
F1.5: Consultar Usuarios

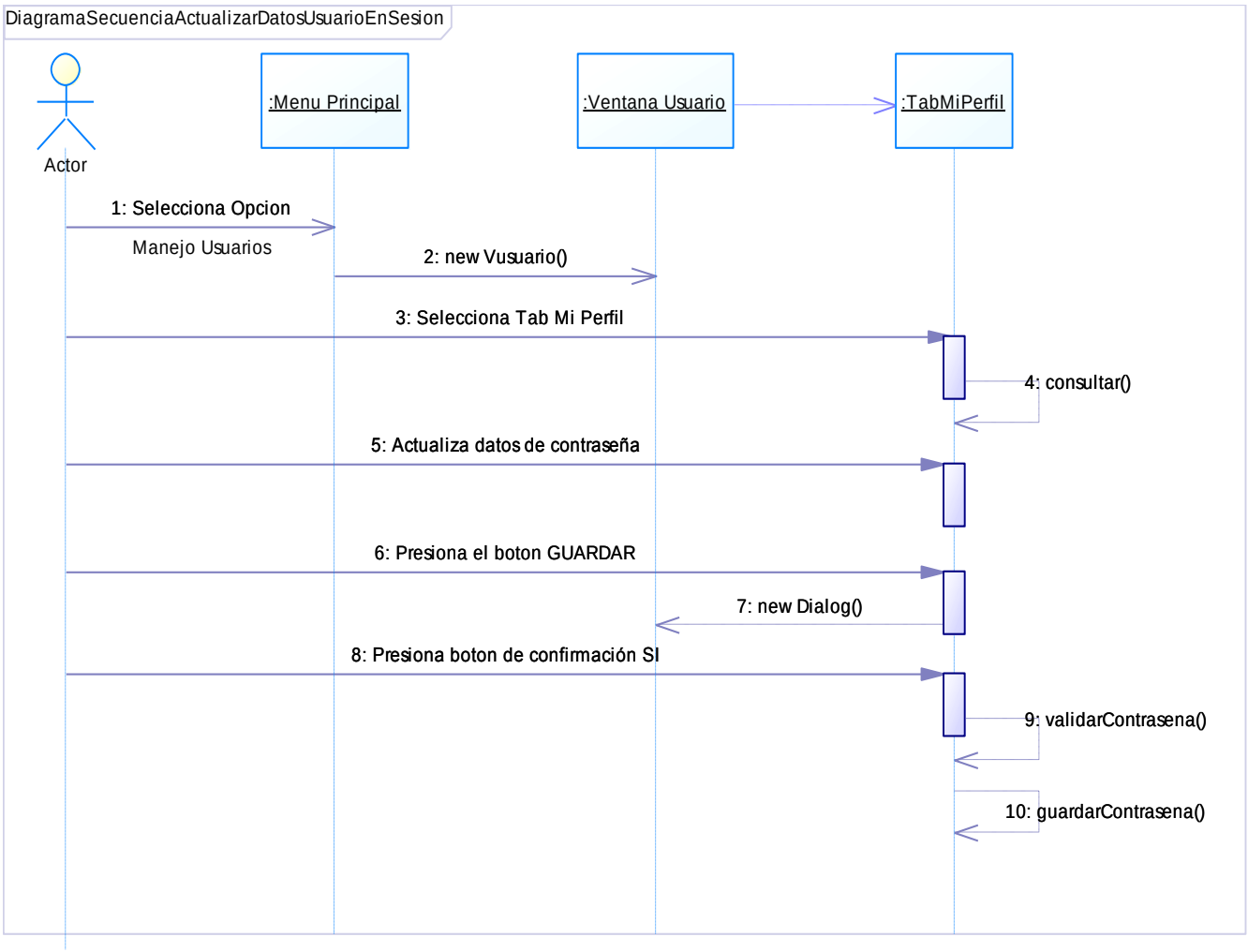


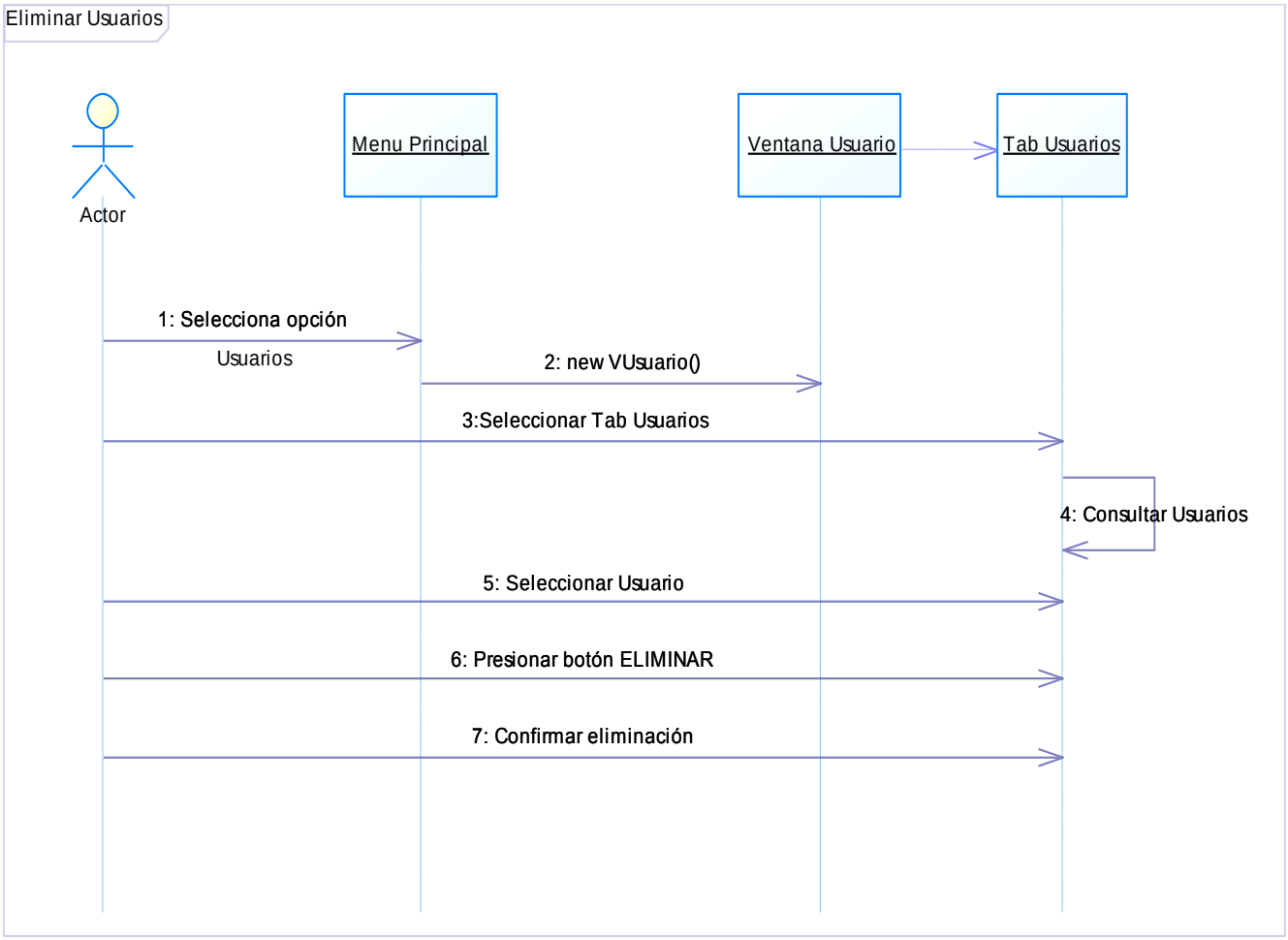
Flujo Principal:

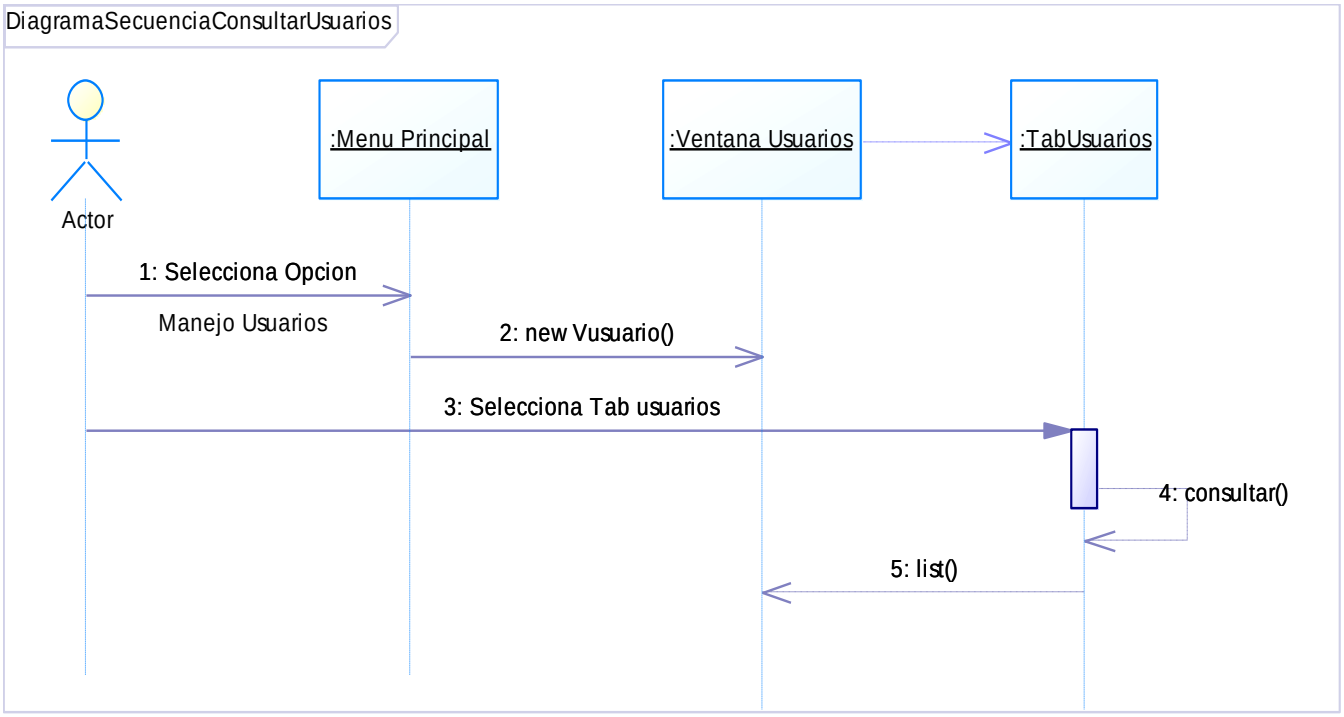
1. El administrador selecciona opción de Administrar Usuarios
2. El administrador selecciona opción de Consultar Usuarios
3. El sistema presenta la ventana con un listado de los usuarios ingresados

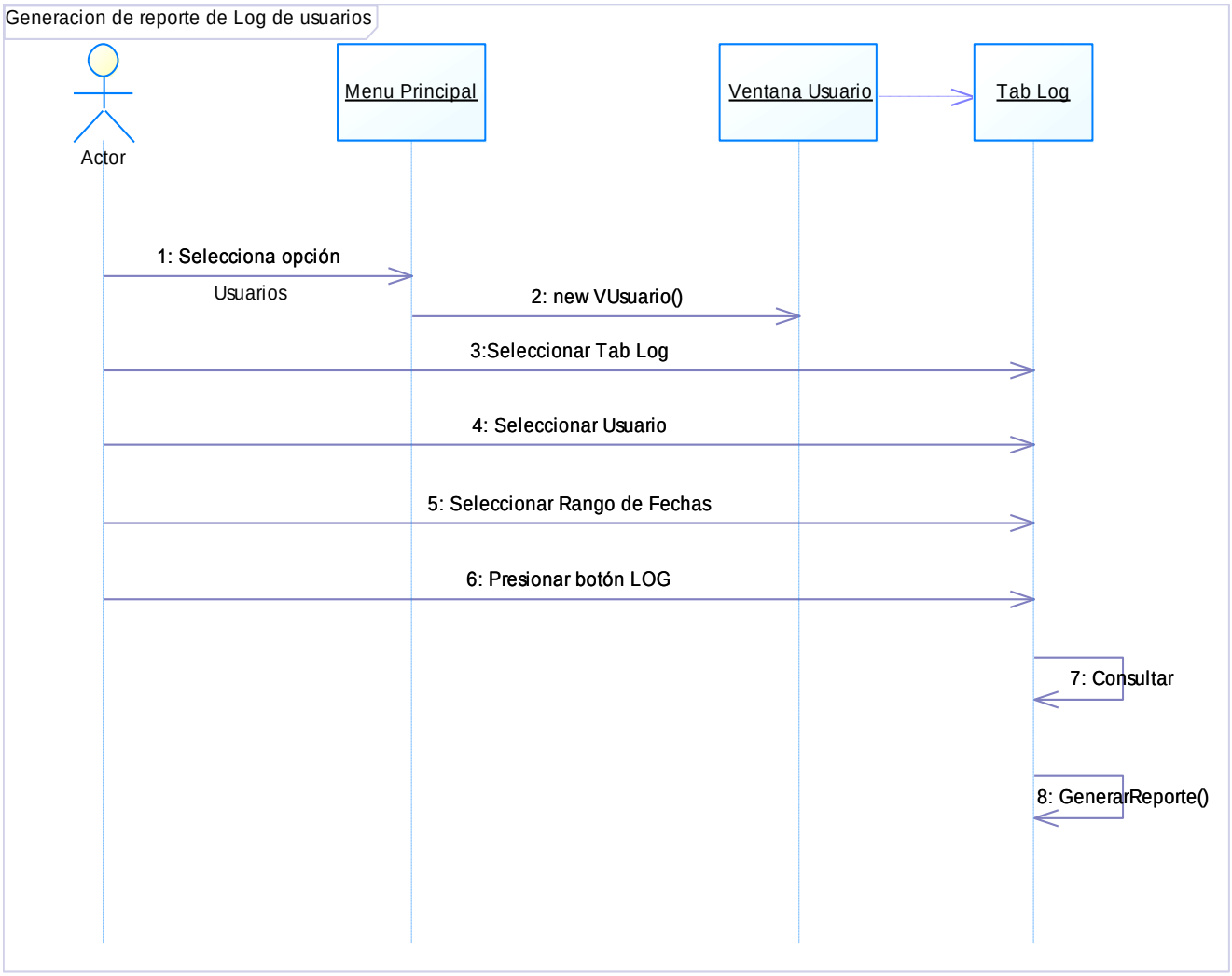

DiagramaSecuenciaCrearUsuario
Actor
:Menu Principal
:Ventana Usuario
:TabCreacionUsuarios
1: Selecciona Opcion
Manejo Usuarios
2: new Vusuario()
3: Selecciona Tab CreacionUsuarios
4: Ingresa nombre del nuevo usuario a crear
5: verificaExistencia()
6: Selecciona perfil de usuario a asignar
7: Ingresa contraseña y verificacion de la misma
8: verificaContrasena()
9: Presiona boton GUARDAR

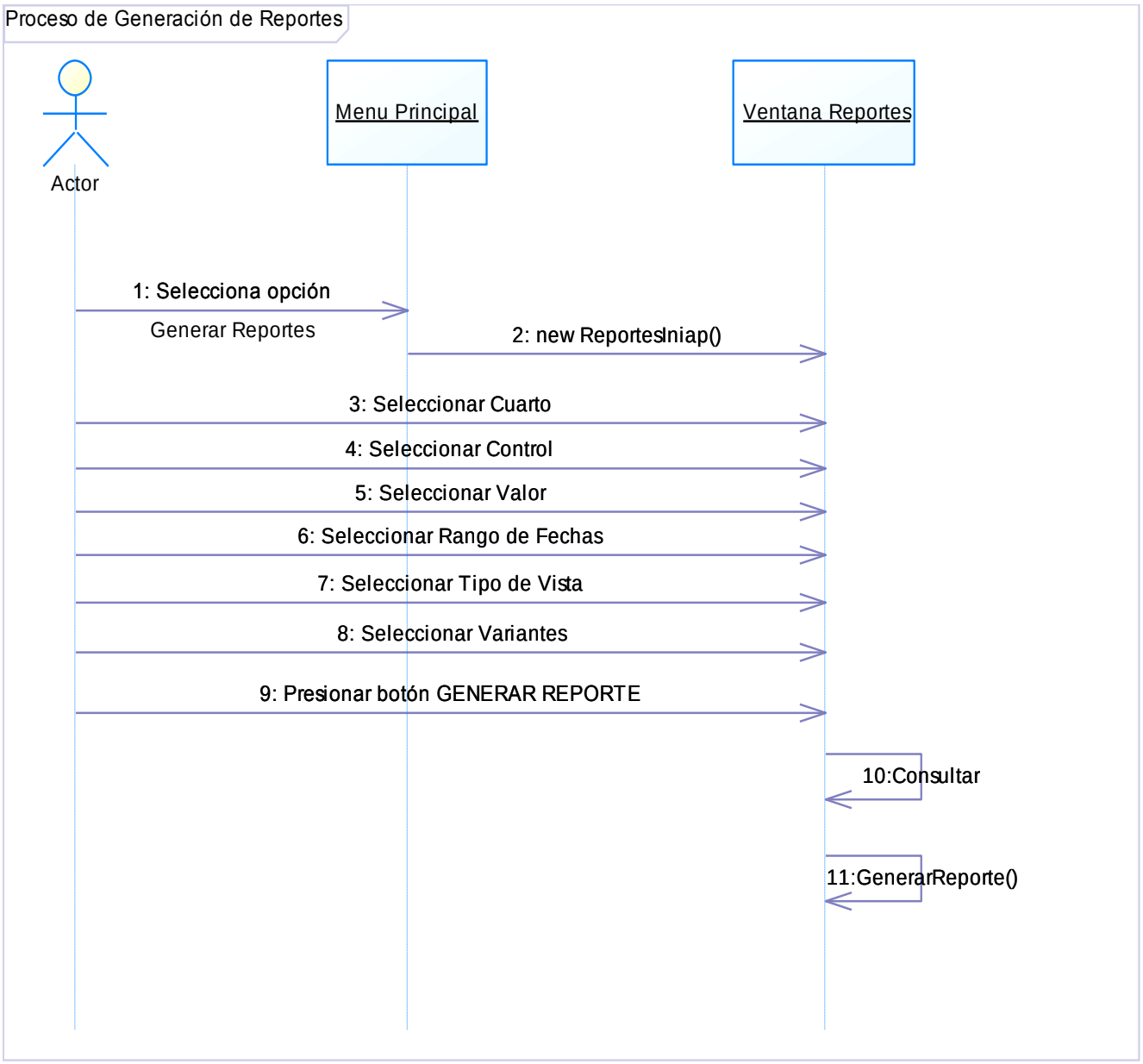

DiagramaSecuenciaActualizarDatos
Actor
:Menu Principal
:Ventana Usuario
:TabUsuarios
1: Selecciona Opcion
Manejo Usuarios
2: new Vusuario()
3: Selecciona Tab Usuarios
4: consultar()
5: Selecciona usuario a modificar datos
6: Presiona boton RESET CONTRASEÑA
7: new Dialog()
8: Presiona boton de confirmación SI
9: reasignarContrasena()


DiagramaSecuenciaActualizarDatosUsuarioEnSesion
Actor
:Menu Principal
:Ventana Usuario
:TabMiPerfil
1: Selecciona Opcion
Manejo Usuarios
2: new Vusuario()
3: Selecciona Tab Mi Perfil
4: consultar()
5: Actualiza datos de contraseña
6: Presiona el boton GUARDAR
7: new Dialog()
8: Presiona boton de confirmación SI
9: validarContrasena()
10: guardarContrasena()


Eliminar Usuarios
Actor
Menu Principal
Ventana Usuario
Tab Usuarios
1: Selecciona opción
Usuarios
2: new VUsuario()
3:Seleccionar Tab Usuarios
4: Consultar Usuarios
5: Seleccionar Usuario
6: Presionar botón ELIMINAR
7: Confirmar eliminación

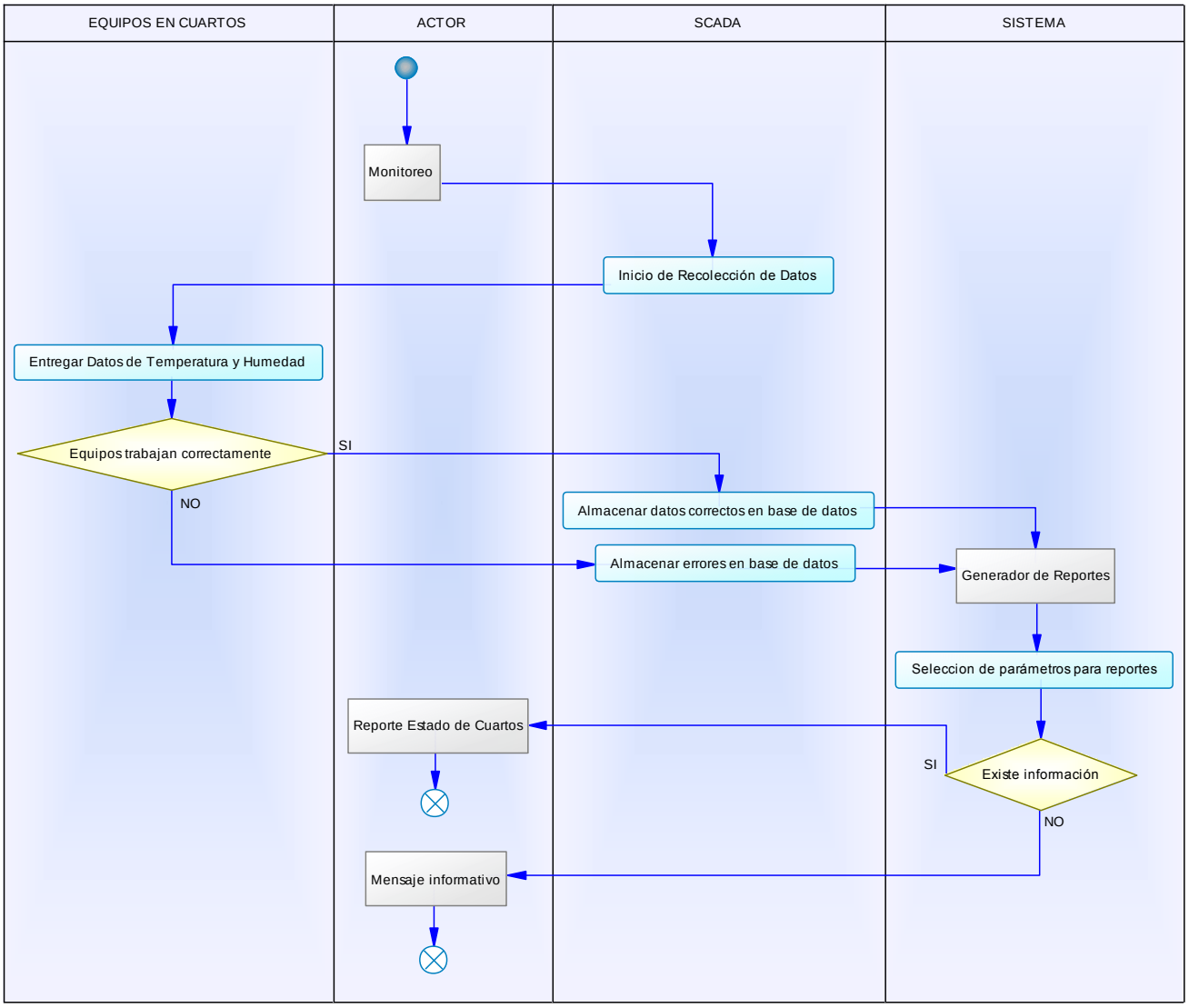

DiagramaSecuenciaConsultarUsuarios
Actor
:Menu Principal
:Ventana Usuarios
:TabUsuarios
1: Selecciona Opcion
Manejo Usuarios
2: new Vusuario()
3: Selecciona Tab usuarios
4: consultar()
5: list()


Generacion de reporte de Log de usuarios
Actor
Menu Principal
Ventana Usuario
Tab Log
1: Selecciona opción
Usuarios
2: new VUsuario()
3:Seleccionar Tab Log
4: Seleccionar Usuario
5: Seleccionar Rango de Fechas
6: Presionar botón LOG
7: Consultar
8: GenerarReporte()


Proceso de Generación de Reportes
Actor
Menu Principal
Ventana Reportes
1: Selecciona opción
Generar Reportes
2: new ReportesIniap()
3: Seleccionar Cuarto
4: Seleccionar Control
5: Seleccionar Valor
6: Seleccionar Rango de Fechas
7: Seleccionar Tipo de Vista
8: Seleccionar Variantes
9: Presionar botón GENERAR REPORTE
10:Consultar
11:GenerarReporte()

4.3 Diagramas de Actividades

En el siguiente diagrama de actividades se observa el procedimiento a seguir para la generación de reportes del estado de la temperatura y humedad de los cuartos, se presenta la interacción entre los equipos instalados en los diferentes cuartos, el actor que es el administrador o el usuario del sistema, el sistema electrónico SCADA y el sistema en sí.



4.4 Historias de Usuario

A continuación se presentan las historias de usuarios desarrolladas conjuntamente con el usuario final. Se utilizó una plantilla originalmente en inglés, obtenida de www.BetterProjects.net, desarrollada por Craig Brown y modificada por los autores para ser utilizada en el desarrollado del presente producto.

Criterios de Aceptación	Ingreso correcto al sistema	
Dado	Información de usuario y contraseña	
Cuando	El usuario y contraseña sean válidos	
Entonces	Se ingresa correctamente al sistema	
	[Back of card]	
Descripción de la Historia	Administración de Usuario	*Prioridad* ___alta___
En el rol de	Administrador del sistema	

Yo quiero/necesito	Tener un control completo para crear, modificar y eliminar usuarios del sistema
	z
Con el objetivo de	Solo una persona pueda tener el control sobre la creación y la información de los usuarios.

[front of card]

Criterios de Aceptación	Debe existir solo un administrador del sistema para controlar los usuarios
Dado	Información del usuario
Cuando	Se verifica si existe o no el usuario
Entonces	Se puede ingresar, modificar o eliminar según sea el caso.

[Back of card]

Descripción de la Historia	Roles de usuario	*Prioridad* ___normal___
En el rol de	Administrador del sistema	
Yo quiero/necesito	Contar con dos tipos de roles de usuario: administrador y usuario general	
Con el objetivo de	Dar permisos a los usuarios para las funciones del sistema según su rol	

[front of card]

Criterios de Aceptación	Debe existir solo un administrador del sistema para controlar los usuarios

Dado	Información del usuario
Cuando	Tenga Asignado el rol respectivo
Entonces	Puede acceder solo a las funciones permitidas para su rol.

[Back of card]

Descripción de la Historia	Enlaces a la base de datos	*Prioridad* ___normal___
En el rol de	Usuario Final	
Yo quiero/necesito	Que el sistema brinde la opción de ingresar la información de la base de datos que se necesite utilizar	
Con el objetivo de	Que la información de la base de datos en el sistema	

	no sea fija, por si se realiza algún cambio de usuario, contraseña, etc en la base.
	[front of card]
Criterios de Aceptación	Los usuarios que no sean "administrador" solo pueden generar su propio log de usuario.
Dado	Datos del usuario en sesión
Cuando	El usuario realice alguna acción en el sistema incluido entrada y salida del mismo.
Entonces	Se registran las acciones con el nombre de usuario, acción realizada, fecha y hora.
	[Back of card]

Descripción de la Historia	Generación de Reportes de monitoreo de cuartos (Temperatura y humedad)	*Prioridad* ___alta___
En el rol de	Usuario Final	

Yo quiero/necesito	Generar reportes del monitoreo realizado en cada cuarto para temperatura (1,2,Promedio) y humedad (1,2,Promedio)
Con el objetivo de	Contar con una ayuda informativa e histórica del monitoreo para la toma de decisiones.

[front of card]

Criterios de Aceptación	Permitir selección de parámetros para generar el reporte necesario.
Dado	Parámetros de: • 1 Cuarto: Almacenamiento (1,2), Germinación (1,2,3), Isotérmico (1,2,3,4) • 1 control: Temperatura o Humedad • 1 o 2 Valores: Temperatura (1,2,Promedio) o Humedad (1,2,Promedio) • Rango de fechas • Opciones de vista: completa o con variantes.
Cuando	Exista información para los parámetros seleccionados.

Entonces	Generación satisfactoria del reporte.

[Back of card]

Descripción de la Historia	Variantes de tiempo para generación de reportes de monitoreo en cuartos.	*Prioridad* ___normal___
En el rol de	Usuario Final	
Yo quiero/necesito	Que se brinde opciones de variaciones de tiempo para los rangos de fechas entre: 30 min, 1 hora, 2 horas.	
Con el objetivo de	Obtener un número menor de registros en los reportes generados.	

[front of card]

Criterios de Aceptación	Selección de vista por variantes en rangos de fecha
Dado	Selección de una variante de: 30 min, 1 hora, 2 horas
Cuando,	Exista información para parámetros seleccionados

Entonces,	Generar reporte con filtros de parámetros y con variante seleccionada para el rango de fechas establecido.

[Back of card]

Descripción de la Historia	Generación de Reportes de monitoreo de cuartos (Temperatura y humedad)	*Prioridad* ___alta___
En el rol de	Usuario Final	
Yo quiero/necesito	Generar reportes del monitoreo realizado en cada cuarto para temperatura (1,2,Promedio) y humedad (1,2,Promedio)	
Con el objetivo de	Contar con una ayuda informativa e histórica del monitoreo para la toma de decisiones.	
	[front of card]	
Criterios de Aceptación	Permitir selección de parámetros para generar el reporte necesario.	
Dado	Parámetros de: • 1 Cuarto: Almacenamiento (1,2), Germinación (1,2,3), Isotérmico (1,2,3,4) • 1 control: Temperatura o Humedad • 1 o 2 Valores: Temperatura (1,2,Promedio) o Humedad (1,2,Promedio)	

	• Rango de fechas • Opciones de vista: completa o con variantes.
Cuando	Exista información para los parámetros seleccionados.
Entonces	Generación satisfactoria del reporte.
	[Back of card]

Descripción de la Historia	Generación de reportes de aviso dependientes del tipo de cuarto en un rango de fechas determinado.	*Prioridad* ___alta___
En el rol de	Usuario Final	
Yo quiero/necesito	Que se brinde opciones del tipo de aviso deseado dentro del rango de fechas seleccionado.	
Con el objetivo de	Visualizar cuando se activan o desactivan equipos y cuando hay fallas en la temperatura o humedad.	
	[front of card]	

Criterios de Aceptación	Selección de parámetros según el tipo de aviso seleccionado.

Dado	El tipo de aviso y los parámetros de tiempo
Cuando,	Exista información para parámetros seleccionados
Entonces,	Generar reporte con filtros de parámetros y con variante seleccionada para el rango de fechas establecido.

[Back of card]

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5. CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- La aplicación de la metodología de extreme programming en el desarrollo del sistema de monitoreo hizo evidente la importancia de contar con el usuario final como parte del equipo de trabajo para mantenerlo al tanto de los avances, nuevas funcionalidades y cambios realizados. Al contrario de retrasar el desarrollo o convertirse en una molestia para el resto de miembros del equipo, resulta muy útil su colaboración al dar opiniones desde un punto de vista diferente al resto, para que el desarrollo se enfoque en lo que el usuario realmente requiere.

- La técnica de pair programming resultó beneficiosa en este desarrollo puesto que los 2 programadores nos involucramos en el desarrollo de cada una de las funcionalidades del sistema, al tomar turnos en la codificación encontramos la manera más viable de resolver los problemas y de cumplir con los requerimientos.

- La entrega de varias versiones, generó presión en el desarrollo, para cada entrega se debía contar con una nueva funcionalidad o con alguna solución a algún error encontrado pero que siempre sea visible al usuario, lo cual requirió de dedicación y de una rápida toma de decisiones para cumplir con los tiempos de entrega de las versiones.

- La implantación del sistema de monitoreo, al proporcionar diagramas y datos estadísticos de cambios que se dan en temperatura y humedad durante el proceso de cultivo y germinación de palma Africana, facilita a la institución, realizar análisis de producción y determinar específicamente donde se dio el problema, la hora exacta en el día en el que se dio y a cargo de quien se encontraba el proceso. Lo que ayuda a la directiva de la institución a tomar decisiones respecto a la forma en como se esta llevando el control de los diferentes procesos.

- Las historias de usuario fueron de mucha utilidad en el proceso de desarrollo porque se detalla junto con el usuario el requerimiento con su respectiva prioridad, se pudo llevar un control ordenado de los requerimientos desarrollados y por desarrollar los cuales servían de sustento en las revisiones y pruebas que se realizaban en cada una de las versiones.

- Para realizar el desarrollo de este proyecto satisfactoriamente fue necesario conocer el proceso que se lleva a cabo en la sede de Santo Domingo con la palma africana, para entender de mejor manera los requerimientos y los resultados esperados por parte de los usuarios. Tuvimos que aprender de manera general sobre el funcionamiento de los equipos instalados en los diferentes cuartos así como también del sistema electrónico SCADA que maneja dichos equipos y que almacena la información en la base de datos.

RECOMENDACIONES

- Con los reportes que genera el sistema de monitoreo, seria recomendable incluir indicadores estadísticos como varianza, desviación estándar, promedios, etc, los cuales ayudarán en la toma de decisiones tanto en el proceso de conservación como en la administración y control del personal.

- Una vez completado e implementado el sistema de monitoreo, es recomendable con una proyección a un futuro inmediato, el desarrollo de un sistema de mensajería, que a través de e-mails y mensajes escritos, informe a los funcionarios correspondientes sobre las alertas emitidas en los cuartos de germinación, almacenamiento e isotérmicos. Para la implementación de este último, la mejor opción es integrarlo con el anterior.

- El sistema de monitoreo desarrollado en el presente proyecto tiene una interacción con el sistema SCADA al utilizar los datos almacenados por el mismo en una base de datos, este tema debería ser tomado en cuenta en alguna de las asignaturas de la carrera de sistemas puesto que es muy común la interacción entre sistemas, en este caso fue con un sistema electrónico pero existe una variedad muy amplia de casos que merecen ser estudiados y discutir sobre la manera correcta de realizar dicha integración.

- Por el momento en el departamento de conservación se cuenta únicamente con un equipo que es utilizado para operar el sistema de monitoreo por eso se pidió desarrollar un sistema de escritorio, en el futuro se podría dar la necesidad de que el sistema sea operado por varios usuarios a la vez en diferentes ordenadores para lo cual se podría migrar a una aplicación web.

6. BIBLIOGRAFÍA

- http://2.bp.blogspot.com/_RoP0c8KoLrg/TK1R8K4uh2I/AAAAAAAAADM/oOVkPzEqvLc/s1600/LENG.jpg, 2011
- http://hamd.galeon.com
- http://java.ciberaula.com/articulo/tecnologia_orientada_objetos/
- http://java.ciberaula.com/articulo/tecnologia_orientada_objetos/
- http://java.ciberaula.com/articulo/tecnologia_orientada_objetos/
- http://thales.cica.es/files/glinex/practicas-glinex05/informatica/uml/practica.pdf
- http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html
- http://www.docirs.cl/uml.htm
- http://www.extremeprogramming.org/
- http://www.extremeprogramming.org/
- http://www.galeon.com/hamd/pdf/scada.pdf
- http://www.hipertexto.info/documentos/uml.htm
- http://www.hipertexto.info/documentos/uml.htm, 2011
- http://www.infor.uva.es/~jmrr/2007/TGP/DOC/JAVA.html
- http://www.microsoft.com/spain/sql/productinfo/overview/default.mspx