Modelado y propuesta de Mejora de Procesos de Desarrollo para un entorno académico

ANEXOS

Gineth Andrea López Hoyos

Monografía para optar al título de

Ingeniero de Sistemas

Director: Ing. Pablo Augusto Magé Imbachí

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Sistemas – Grupo de Investigación IDIS

Línea de Investigación Calidad del Software: Producto y Proceso

Popayán, Febrero de 2013

TABLA DE CONTENIDO

ANEXO A

Proceso de ejecución del curso de sistemas distribuidos

Descripción General

Brindar al estudiante bases conceptuales y metodológicas para el desarrollo de Software de Sistemas Distribuidos involucrando las diferentes tecnologías utilizadas en esta área.

Propósito

Se desarrollaran clases magistrales orientadas por el profesor; investigaciones, exposiciones y desarrollo de trabajos prácticos por parte de los alumnos en el transcurso del semestre.

Objetivos

Al final del curso los estudiantes:

- Tendrán las bases conceptuales y metodológicas para el desarrollo de aplicaciones en sistemas distribuidos.
- Dispondrán de las nociones más importantes en cuanto a las tecnologías de programación distribuida, diferenciando con claridad las ventajas e inconvenientes de cada una a la hora de ser aplicadas.
- Conocerán las nociones de programación distribuida utilizando la tecnología RPC, RMI, CORBA y el lenguaje de programación Java, logrando desarrollar aplicaciones distribuidas multiplataforma con Java.

Fases

En esta fase se define cada uno de los elementos que harán parte del curso de laboratorio de sistemas distribuidos, de manera que se puedan cumplir los objetivos establecidos por el programa.

Recopilando, seleccionando y adaptando el material que soporte la temática que se desea abordar en este curso.

De la misma manera se debe especificar la metodología que se va a seguir, en la que se tendrá en cuenta el tiempo que se utilizará abordando cada una de las temáticas, al igual que la profundidad con la que estas estarán siendo abordadas en el curso.

La estrategia de evaluación será definida en esta fase, teniendo en cuenta la estructura que se ha establecido para la realización de éste.

Propósito

Estructurar el plan con el que se va a llevar a cabo la ejecución del curso de laboratorio de sistemas distribuidos, estableciendo claramente el proceso que se debe seguir para conseguir los mejores resultados en el desarrollo de éste.

Objetivos

Definir todos los elementos que son necesarios en el desarrollo del curso de sistemas distribuidos.

Notas Introductorias

Para determinar el estado de las actividades de preparación de la ejecución del curso se debe medir y analizar:

1. Prácticas (Fase Iterativa)

Descripción General

Esta fase tiene como finalidad el desarrollo de los conceptos que se han impartido a lo largo del curso de sistemas distribuidos, de manera que los estudiantes puedan apropiar cada uno de los conceptos.

La primera parte de esta fase corresponde a la preparación del desarrollo de estas prácticas, recopilando, seleccionando y adaptando el material que soporte la temática que se desea abordar en este curso. De la misma manera se debe especificar la metodología que se va a seguir, en la que se tendrá en cuenta el tiempo que se utilizará abordando cada una de las temáticas, al igual que la profundidad con la que estas estarán siendo abordadas en el curso.

La estrategia de evaluación será definida en esta fase, teniendo en cuenta la estructura que se ha establecido para la realización de éste.

La segunda instancia se enfoca en la realización prácticas de desarrollo de las guías desarrolladas en la preparación de manera que se apropien los conocimientos impartidos en el curso de sistemas distribuidos.

El profesor entrega a los estudiantes la guía que debe realizarse, y se inicia el proceso de desarrollo asociado a la solución del problema propuesto en ésta.

El estudiante en su capacidad de ejecutor se encarga de la solución del problema, guiado por el profesor en su capacidad de supervisor, para que la práctica sea realizada en forma optima.

Finalmente, el profesor en su capacidad de evaluador revisa cada uno de las fuentes entregados por los estudiantes después de la realización de la práctica, para corroborar que el concepto haya sido asimilado de manera efectiva.

Esta revisión esta basada en la guía que se ha entregado anteriormente al estudiante, de manera que se cumpla con los puntos establecidos en la misma, evitando de esta manera la omisión de alguno de los elementos señalados.

Propósito

A través de la resolución de un problema de programación identificar las falencias conceptuales que los estudiantes del curso de Sistemas Distribuidos puedan tener acerca del concepto que se esta abordando.

Estructurar el plan con el que se va a llevar a cabo la ejecución del curso de laboratorio de sistemas distribuidos, estableciendo claramente el proceso que se debe seguir para conseguir los mejores resultados en el desarrollo de éste.

Objetivos

Definir todos los elementos que son necesarios en el desarrollo del curso de sistemas distribuidos.

Apropiar cada uno de los conceptos que se han impartido en el curso de Sistemas Distribuidos, resolviendo un problema de programación que se ha adaptado a las necesidades del concepto que se desea abordar.

Analizar como cada uno de los estudiantes abordo el problema, y si éste fue resuelto en el tiempo que se le asigno, verificando del mismo modo la apropiación del concepto en cada caso.

2. Proyectos (Fase Iterativa)

Descripción General

Esta fase los grupos de trabajo definidos con anterioridad, deben dar solución al requerimiento que se ha definido para la tecnología con la que se esta trabajando.

Este trabajo tiene dos tipos de entregables que serán desarrollados por los grupos.

De la misma manera que se debe realizar la aplicación que de solución al problema planteado, se deben generar tres manuales que soporten el desarrollo de la aplicación.

Los elementos que serán plasmados en éstos, son definidos por el profesor en la fase de preparación, por lo cual se espera una concordancia de nivel alto entre el desarrollo que se presenta y los manuales entregados.

Propósito

Afianzar los conocimientos adquiridos durante las prácticas que se desarrollaron con anterioridad, de manera que se puedan plasmar en la óptima solución del requerimiento especificado. (Revisar)

Objetivos

Desarrollar una aplicación en el ámbito de los Sistemas Distribuidos, que cumpla con los requerimientos que se definieron, de manera que se apliquen todos y cada uno de los conceptos que se han ido abordando en el curso.

3. Cierre (Fase)

Descripción General

El profesor en su capacidad de evaluador revisa los elementos entregados por los estudiantes en el plazo que se ha definido para el desarrollo del requerimiento.

Esta revisión está basada en la guía que se ha entregado anteriormente al estudiante, de manera que se cumpla con los puntos establecidos en la misma, evitando de esta manera la omisión de alguno de los elementos señalados.

Debido a que este desarrollo se realiza fuera del ambiente de clase, se debe realizar también una sustentación en la que se definirá si el estudiante en verdad realizó o no el requerimiento que se le ha encargado.

La fase de revisión aquí señalada, tiene dos partes, en las que el profesor en su calidad de evaluador, realiza diversos roles que le permiten definir si el requerimiento desarrollado cumple con las restricciones definidas:

- Revisión de la aplicación.
- Revisión de los manuales asociados a la aplicación.

Esta revisión es compacta de manera que en el momento en que se revisan los fuentes se usan con frecuencia los manuales que se han entregado debido a que se desea comprobar la concordancia de las éstos.

Propósito

Revisar la manera en la que los estudiantes resuelven el problema descrito, y si están o no en capacidad de manejar los conceptos que se evaluaron durante la práctica.

Objetivos

Analizar como cada uno de los estudiantes abordo el problema, y si éste fue resuelto en el tiempo que se le asigno, verificando del mismo modo la apropiación del concepto en cada caso.

FASE PRÁCTICAS

1. **Actividad: Preparar Sala**

1.1 Tarea: Reportar Equipos

Descripción	Se realiza un listado de los equipos que están disponibles en la sala en la que se llevaran a cabo las prácticas.
Entradas	
Salidas	Reporte de Equipos
Roles	Profesor , Monitor

1.2 Tarea: Instalar Herramientas

Descripción	Se realiza la instalación de cada una de las herramientas requeridas para la correcta ejecución de las prácticas de cada una de las tecnologías que se dictan en el curso.
Entradas	Guía de Instalación de Herramientas
Salidas	Reporte de Equipos
Roles	Profesor , Monitor

1.3 Tarea: Probar Configuración

Descripción	Se prueba la configuración realizada en cada uno de los equipos con una práctica Demo que realizó el docente de la asignatura.
Entradas	Demo Práctica
Salidas	Reporte de Equipos
Roles	Profesor , Monitor

Actividades y tareas

2. Actividad: Preparar Práctica

2.1Tarea: Selección del Problema

Descripción	Se realiza una búsqueda en Internet de un problema de programación para desarrollar el concepto que se quiere abordar.
Entradas	Textos guía, Documento del plan de la asignatura
Salidas	Problema seleccionado
Roles	Profesor (Preparador)

2.2 Tarea: Adaptar concepto Sistemas Distribuidos (SD)

Descripción	Se realiza los cambios necesarios al planteamiento del problema de manera que se pueda a través de este afianzar el concepto seleccionado.
Entradas	Problema seleccionado
Salidas	Problema seleccionado adaptado al concepto de sistemas distribuidos
Roles	Profesor (Preparador)

2.3 Tarea: Implementar prototipo

Descripción	Se elabora la solución del problema, para continuar con el refinamiento del mismo, de modo que sea accesible para los estudiantes y se pueda realizar en el tiempo asignado.
Entradas	Problema seleccionado adaptado al concepto de sistemas distribuidos
Salidas	Prototipo
Roles	Profesor (Preparador)

2.4 Tarea: Elaborar guía

Descripción	Se toma toda la información que se ha almacenado en las actividades anteriores y se escribe la guía que será desarrollada.
Entradas	Fuentes Demo, Problema seleccionado adaptado al concepto de sistemas distribuidos.
Salidas	Guía
Roles	Profesor (Preparador)

2.5 Tarea: Presentar Guía

Descripción	Se realiza la exposición y explicación de la guía que se desarrollará en esa sección de clase.
Entradas	Guía
Salidas	NINGUNA
Roles	Profesor (Orador)

3. Actividad: Ejecutar práctica

3.1 Tarea: Analizar

3.1.1 Paso1: Priorización de requisitos

Descripción	El estudiante realiza una lista en la que selecciona teniendo en cuenta diversos factores, cuáles de los requerimientos especificados en la guía tienen un mayor grado de importancia en la solución del problema.
Entradas	Guía de la práctica i-ésima
Salidas	Entregable de priorización de requisitos
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.1.2 Paso 2: Viabilidad del proyecto

Descripción	El estudiante define como va a usar el tiempo que le ha sido asignado, y de esta forma definir si es o no posible desarrollar a conformidad la práctica.
Entradas	Entregable de priorización de requisitos
Salidas	Entregable de Viabilidad del proyecto
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.1.3 Paso 3: Selección de la arquitectura

Descripción	El estudiante define como se manejará la persistencia para dar solución al problema que se le ha asignado.
Entradas	Entregable de Viabilidad del proyecto
Salidas	Entregable de Selección de Arquitectura
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.1.4 Paso 4: Manejo de persistencia

Descripción	El estudiante define cual es la manera adecuada de guardar la información que es relevante para la solución del problema.
Entradas	Entregable de Análisis
Salidas	Entregable de Manejo de Persistencia
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.2 Tarea: Diseñar

3.2.1 Paso 1: Realizar Diagramas de casos de uso

Descripción	El estudiante realiza los diagramas correspondientes a las acciones que cada uno de los usuarios de la aplicación podrá llevar a cabo una vez finalizado.
Entradas	Entregable de Análisis
Salidas	Entregable Diagramas de Casos de uso
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.2.2 Paso 2: Realizar Diagrama de clases

Descripción	El estudiante realiza los diagramas correspondientes a las entidades que definió para la solución del problema.
Entradas	Entregable de Análisis, Entregable de Diagramas de Casos de uso
Salidas	Entregable Diagrama de Clases
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.2.3 Paso 3: Definir Manejo de persistencia

Descripción	El estudiante define cual es la manera adecuada de guardar la información que es relevante para la solución del problema.
Entradas	Entregable de Análisis
Salidas	Entregable de Manejo de Persistencia
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.2.4 Paso 4. Realizar Diagramas de secuencia

Descripción	El estudiante realiza los diagramas correspondientes al flujo que se debe seguir para conseguir los resultados esperados en la aplicación.
Entradas	Entregable Casos de Uso
Salidas	Entregable Diagramas de Secuencia
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.2.5 Paso 5: Realizar Diagramas de colaboración

Descripción	El estudiante realiza los diagramas correspondientes a la interacción que tienen todos los elementos inmersos en la aplicación.
Entradas	Entregable Casos de Uso
Salidas	Entregable Diagramas de Colaboración
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.3 Tarea: Desarrollar

3.3.1 Paso 1: Programación por pares

Descripción	Los estudiantes desarrollan las líneas de código necesarias para implementar cada una de las funcionalidades que fueron analizadas y diseñadas en las actividades anteriores, esta tarea será realizada en pares para garantizar un código fuente de mayor calidad.
Entradas	Entregable de Diseño
Salidas	Fuentes de Requerimiento
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.3.2 Paso 2: Pruebas unitarias

Descripción	Cada vez que se desarrolla una funcionalidad los estudiantes (desarrolladores) realizan pruebas en las que se pueden encontrar fallos que serán solucionados de manera efectiva por éstos.
Entradas	Fuentes de Requerimiento
Salidas	Reporte de pruebas unitarias
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.4 Tarea: Probar

3.4.1 Paso 1: Pruebas de acoplamiento

Descripción	Se realiza el acople de toda la funcionalidad que se ha desarrollado, y sobre esta se hacen pruebas en las que se observa si la aplicación esta o no cumpliendo con las especificaciones descritas.
Entradas	Entregable de Implementación
Salidas	Reporte de pruebas de acoplamiento
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.4.2 Paso 2: Pruebas de valor límite

Descripción	Estas pruebas tienen como finalidad ver cómo se comporta la aplicación con una serie de entradas específicas, revisando en el peor de los casos la aplicación.
Entradas	Entregable de Implementación
Salidas	Reporte de pruebas de valor límite
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3.5 Tarea: Entregar

Descripción	Esta tarea consiste en acoplar todos los elementos que han sido desarrollados con anterioridad para cumplir con las exigencias especificadas.
Entradas	Entregable de Análisis, Entregable de Diseño, Entregable de Implementación, Entregable de Pruebas.
Salidas	Fuentes Práctica i-ésima, Manual
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

4. Actividad: Preparar Evaluación

4.1 Tarea: Definir Grupos

Descripción	Los estudiantes eligen a su compañero de trabajo para el curso.
Entradas	Lista de Estudiantes
Salidas	Lista de Grupos
Roles	Estudiante (Seleccionador) Profesor (Observador)

4.2 Tarea: Generar Formatos de Evaluación

Descripción	El docente define cada una de las evaluaciones que serán aplicadas a cada uno de las prácticas y requerimientos que serán realizados.
Entradas	
Salidas	Formato Extendido Requerimiento, Formato Extendido Reporte, Formato Extendido Prácticas, Formato Reporte Notas
Roles	Profesor (Evaluador)

Actividades y tareas

5. Actividad: Evaluación de la práctica

5.1 Tarea: Revisión cliente

Descripción	El cliente realiza la ejecución de la aplicación teniendo en cuenta el manual de usuario, de esta manera podrá encontrar los fallos. El cliente experto realiza la compilación, teniendo en cuenta el manual de instalación proporcionado, así podrá verificar si el código fuente ha sido generado satisfactoriamente y si existe concordancia con lo que se especifico en el manual. Paso siguiente, se ejecutará la aplicación, para realizar un chequeo de las funcionalidades que se especificaron, cuantificando cada uno de estos para obtener los reportes correspondientes.
Entradas	Fuentes del requerimiento
Salidas	Reporte de fallos de cliente
Roles	Profesor(Evaluador)
Observaciones	En el caso de esta fase esta tarea se realiza a un nivel micro, debido a que el problema que se seleccionó es pequeño y realizado en poco tiempo. No existen manuales de manera que esta revisión será mucho más puntual.

5.2 Tarea: Revisión Diseñador

Descripción	El diseñador hace una inspección en la que tiene en cuenta el código fuente de la aplicación y el manual técnico en el que se han especificado los elementos que se desarrollaran, de manera que estos concuerden con los que se están entregando. De esta manera el diseñador se dispone a revisar como los estudiantes resolvieron el problema, y cuáles fueron las decisiones que tomaron para esta tarea. Así pues se evaluará la manera en la que los estudiantes analizaron, diseñaron e implementaron el código. Teniendo como meta que estas actividades hayan sido realizadas secuencialmente y que las decisiones que se tomaron anteriormente sean la base de las decisiones de las etapas posteriores, de modo que haya secuencia en éstas.
Entradas	Fuentes del requerimiento
Salidas	Reporte de fallos diseñador
Roles	Profesor(Evaluador)
Observaciones	En el caso de esta fase esta tarea se realiza a un nivel micro, debido a que el problema que se seleccionó es pequeño y realizado en poco tiempo. No existen manuales de manera que esta revisión será mucho más puntual.

5.4 Tarea: Realizar reporte de fallos

Descripción	Se reúnen los reportes desarrollados en las tareas anteriores y se escribe el reporte final de fallos, en el que se encuentran la descripción de los fallos y la cuantificación de la práctica.
Entradas	Reporte de fallos de cliente, Reporte de fallos diseñador
Salidas	Reporte de Fallos Práctica
Roles	Profesor(Evaluador)

FASE PROYECTOS

Actividades y tareas

1. **Actividad: Preparar Proyecto**

1.1 Tarea: Selección del Problema

Descripción	Se realiza una búsqueda en Internet de un problema de programación para desarrollar el concepto que se quiere abordar.
Entradas	Textos guía, Documento del plan de la asignatura
Salidas	Problema seleccionado
Roles	Profesor (Preparador)

1.2 Tarea: Adaptar concepto Sistemas Distribuidos (SD)

Descripción	Se realiza los cambios necesarios al planteamiento del problema de manera que se pueda a través de este afianzar el concepto seleccionado.
Entradas	Problema seleccionado
Salidas	Problema seleccionado adaptado al concepto de sistemas distribuidos
Roles	Profesor (Preparador)

1.3 Tarea: Implementar prototipo

Descripción	Se elabora la solución del problema, para continuar con el refinamiento del mismo, de modo que sea accesible para los estudiantes y se pueda realizar en el tiempo asignado.
Entradas	Problema seleccionado adaptado al concepto de sistemas distribuidos
Salidas	Prototipo
Roles	Profesor (Preparador)

1.4 Tarea: Elaborar guía

Descripción	Se toma toda la información que se ha almacenado en las actividades anteriores y se escribe la guía que será desarrollada.
Entradas	Fuentes Demo, Problema seleccionado adaptado al concepto de sistemas distribuidos.
Salidas	Guía
Roles	Profesor (Preparador)

1.5 Presentar Guía

Descripción	Se realiza la exposición y explicación de la guía que se desarrollará en esa sección de clase.
Entradas	Guía
Salidas	NINGUNA
Roles	Profesor (Orador)

Actividades y tareas

2. Actividad: Ejecutar proyecto

2.1 Tarea: Analizar

2.1.1 Paso1: Priorización de requisitos

Descripción	El estudiante realiza una lista en la que selecciona teniendo en cuenta diversos factores, cuáles de los requerimientos especificados en la guía tienen un mayor grado de importancia en la solución del problema.
Entradas	Guía de la práctica i-ésima
Salidas	Entregable de priorización de requisitos
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.1.2 Paso 2: Viabilidad del proyecto

Descripción	El estudiante define como va a usar el tiempo que le ha sido asignado, y de esta forma definir si es o no posible desarrollar a conformidad la práctica.
Entradas	Entregable de priorización de requisitos
Salidas	Entregable de Viabilidad del proyecto
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.1.3 Paso 3: Selección de la arquitectura

Descripción	El estudiante define como se manejará la persistencia para dar solución al problema que se le ha asignado.
Entradas	Entregable de Viabilidad del proyecto
Salidas	Entregable de Selección de Arquitectura
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.1.4 Paso 4: Manejo de persistencia

Descripción	El estudiante define cual es la manera adecuada de guardar la información que es relevante para la solución del problema.
Entradas	Entregable de Análisis
Salidas	Entregable de Manejo de Persistencia
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.2 Tarea: Diseñar

2.2.1 Paso 1: Realizar Diagramas de casos de uso

Descripción	El estudiante realiza los diagramas correspondientes a las acciones que cada uno de los usuarios de la aplicación podrá llevar a cabo una vez finalizado.
Entradas	Entregable de Análisis
Salidas	Entregable Diagramas de Casos de uso
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.2.2 Paso 2: Realizar Diagrama de clases

Descripción	El estudiante realiza los diagramas correspondientes a las entidades que definió para la solución del problema.
Entradas	Entregable de Análisis, Entregable de Diagramas de Casos de uso
Salidas	Entregable Diagrama de Clases
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.2.3 Paso 3: Manejo de persistencia

Descripción	El estudiante define cual es la manera adecuada de guardar la información que es relevante para la solución del problema.
Entradas	Entregable de Análisis
Salidas	Entregable de Manejo de Persistencia
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.2.4 Paso 4: Realizar Diagramas de secuencia

Descripción	El estudiante realiza los diagramas correspondientes al flujo que se debe seguir para conseguir los resultados esperados en la aplicación.
Entradas	Entregable Casos de Uso
Salidas	Entregable Diagramas de Secuencia
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.2.5 Paso 5: Realizar Diagramas de colaboración

Descripción	El estudiante realiza los diagramas correspondientes a la interacción que tienen todos los elementos inmersos en la aplicación.
Entradas	Entregable Casos de Uso
Salidas	Entregable Diagramas de Colaboración
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.3 Tarea: Desarrollar

2.3.1 Paso 1: Programación por pares

Descripción	Los estudiantes desarrollan las líneas de código necesarias para implementar cada una de las funcionalidades que fueron analizadas y diseñadas en las actividades anteriores, esta tarea será realizada en pares para garantizar un código fuente de mayor calidad.
Entradas	Entregable de Diseño
Salidas	Fuentes de Requerimiento
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.3.2 Paso 2: Pruebas unitarias

Descripción	Cada vez que se desarrolla una funcionalidad los estudiantes (desarrolladores) realizan pruebas en las que se pueden encontrar fallos que serán solucionados de manera efectiva por éstos.
Entradas	Fuentes de Requerimiento
Salidas	Reporte de pruebas unitarias
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.4 Tarea: Probar

2.4.1 Paso 1: Pruebas de acoplamiento

Descripción	Se realiza el acople de toda la funcionalidad que se ha desarrollado, y sobre esta se hacen pruebas en las que se observa si la aplicación esta o no cumpliendo con las especificaciones descritas.
Entradas	Entregable de Implementación
Salidas	Reporte de pruebas de acoplamiento
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.4.2 Paso 2: Pruebas de valor límite

Descripción	Estas pruebas tienen como finalidad ver cómo se comporta la aplicación con una serie de entradas especificas, revisando en el peor de los casos la aplicación.
Entradas	Entregable de Implementación
Salidas	Reporte de pruebas de valor límite
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

2.5 Entregar

Descripción	Esta tarea consiste en acoplar todos los elementos que han sido desarrollados con anterioridad para cumplir con las exigencias especificadas.
Entradas	Entregable de Análisis, Entregable de Diseño, Entregable de Implementación, Entregable de Pruebas.
Salidas	Fuentes Práctica i-ésima, Manual
Roles	Profesor (Supervisor), Estudiante (Ejecutor)

3. Actividad: Preparar Evaluación

3.1 Tarea: Definir Grupos

Descripción	Los estudiantes eligen a su compañero de trabajo para el curso.
Entradas	Lista de Estudiantes
Salidas	Lista de Grupos
Roles	Estudiante (Seleccionador) Profesor (Observador)

3.2 Tarea: Generar Formatos de Evaluación

Descripción	El docente define cada una de las evaluaciones que serán aplicadas a cada uno de las prácticas y requerimientos que serán realizados.
Entradas	
Salidas	Formato Extendido Requerimiento, Formato Extendido Reporte, Formato Extendido Prácticas, Formato Reporte Notas
Roles	Profesor (Evaluador)

3.3 Tarea: Generar Formatos de Sustentación

Descripción	El docente define las preguntas que serán realizadas en la actividad de sustentación que se realiza de manera exclusiva sobre el requerimiento.
Entradas	
Salidas	Formato Sustentación
Roles	Profesor (Evaluador)

4. Actividad: Evaluación del Requerimiento

4.1 Tarea: Revisión cliente

Descripción	El cliente realiza la ejecución de la aplicación teniendo en cuenta el manual de usuario, de esta manera podrá encontrar los fallos. El cliente experto realiza la compilación, teniendo en cuenta el manual de instalación proporcionado, así podrá verificar si el código fuente ha sido generado satisfactoriamente y si existe concordancia con lo que se especifico en el manual. Paso siguiente, se ejecutará la aplicación, para realizar un chequeo de las funcionalidades que se especificaron, cuantificando cada uno de estos para obtener los reportes correspondientes.
Entradas	Fuentes del requerimiento
Salidas	Reporte de fallos de cliente
Roles	Profesor(Evaluador)
Observaciones	En el caso de esta fase esta tarea se realiza a un nivel micro, debido a que el problema que se seleccionó es pequeño y realizado en poco tiempo. No existen manuales de manera que esta revisión será mucho más puntual.

4.2 Tarea: Revisión Diseñador

Descripción	El diseñador hace una inspección en la que tiene en cuenta el código fuente de la aplicación y el manual técnico en el que se han especificado los elementos que se desarrollaran, de manera que estos concuerden con los que se están entregando. De esta manera el diseñador se dispone a revisar como los estudiantes resolvieron el problema, y cuáles fueron las decisiones que tomaron para esta tarea. Así pues se evaluará la manera en la que los estudiantes analizaron, diseñaron e implementaron el código. Teniendo como meta que estas actividades hayan sido realizadas secuencialmente y que las decisiones que se tomaron anteriormente sean la base de las decisiones de las etapas posteriores, de modo que haya secuencia en éstas.
Entradas	Fuentes del requerimiento
Salidas	Reporte de fallos diseñador
Roles	Profesor(Evaluador)
Observaciones	En el caso de esta fase esta tarea se realiza a un nivel micro, debido a que el problema que se seleccionó es pequeño y realizado en poco tiempo. No existen manuales de manera que esta revisión será mucho más puntual.

4.3 Tarea: Realizar reporte de fallos

Descripción	Se reúnen los reportes desarrollados en las tareas anteriores y se escribe el reporte final de fallos, en el que se encuentran la descripción de los fallos y la cuantificación de la práctica.
Entradas	Reporte de fallos de cliente, Reporte de fallos diseñador
Salidas	Reporte de Fallos Práctica
Roles	Profesor(Evaluador)

4.4 Sustentar

Descripción	**Los estudiantes dan solución a las preguntas que se establecen en la guía, y así se establecen las calificaciones pertinentes.**
Entradas	**Guía de Sustentación**
Salidas	**Reporte de Factor de Sustentación**
Roles	**Profesor(Evaluador), Estudiante (Evaluado)**

FASE CIERRE

1. Actividad: Cierre

1.1 Tarea: Ponderar Nota

Descripción	El profesor en su capacidad de evaluador cuantifica las notas que cada uno de los estudiantes recibió en las diferentes actividades realizadas durante todo el proceso.
Entradas	Reporte Cliente, Reporte Diseñador, Reporte Factor de Sustentación
Salidas	Reportes de Notas
Roles	Profesor (Evaluador)

1.2 Tarea: Publicar Reportes

Descripción	El profesor publica en el sitio de la asignatura las notas de cada uno de los estudiantes.
Entradas	Reporte Cliente, Reporte Diseñador, Reporte Factor de Sustentación, Reportes de Notas
Salidas	
Roles	Profesor (Evaluador)

ANEXO B

PLAN DE MEJORA

Proyecto de Implementación
Mejora Continua
Laboratorio de Sistemas Distribuidos

Autores:

Gineth Andrea López Hoyos

TABLA DE CONTENIDO

Plan del Proyecto

1 Introducción

1.1 Propósito

El propósito de este documento es permitir organizar, controlar y administrar la documentación referente al proyecto **Implementación de Mejora Continua** basado en el **Ágile SPI** para el curso de pregrado **Laboratorio de Sistemas Distribuidos.** En este documento se encuentra la referencia a todos los documentos generados durante el proyecto los cuales se irán agregando y complementando a lo largo del proyecto.

1.2 Ámbito

El ámbito de este documento es la reutilización de elementos de **Tutelkán Reference Process** (TRP) en para el curso de pregrado Laboratorio de Sistemas Distribuidos, además de la definición del proceso existente y las adecuaciones necesarias que se encontraron a lo largo de la aplicación de la metodología atendiendo a las necesidades del cliente.

Describir el contexto académico. Reemplazar.

1.3 Definiciones, acrónimos y abreviaciones.

TRP: Tutelkán Reference Process.

IPC: Implementación Proceso Consultores.

ETP: Equipos de Tecnología de Procesos.

LSD: Laboratorio de Sistemas Distribuidos.

SPI: Software Process Impromevement.

QA: Quality Assurance

1.4 Referencias

Ver Capítulo de ANEXOS.

2 Resumen del Proyecto

2.1 Propósito del Proyecto, ámbito, y Objetivos

El propósito de este proyecto es generar la versión inicial del Proceso de Desarrollo del Laboratorio de Sistemas Distribuidos teniendo en cuenta cada una de las restricciones que se deben tener en cuenta debido al tipo de contexto en el que se desenvuelve el proyecto.

Se espera que al término del proyecto se tenga:

- Definición inicial del proceso que se lleva a cabo en el Laboratorio de Sistemas Distribuidos.
- Definición de algunos indicadores del proceso, con los que se pretende la evaluación de la implantación del programa de mejora que se ha definido.
- Propuesta de un plan de mejora de procesos que tiene en cuenta las necesidades expuestas por el docente de la materia y las que se han encontrado a lo largo de la revisión del proceso en el ambiente real.
- Dar pie para un proceso de mejora continua dentro del marco que provee el ámbito académico en el que se desenvuelve el proyecto.

2.2 Estrategia Global del proyecto

Este proyecto se está llevando a cabo realizando la definición inicial de cada uno de los elementos que harán parte del proceso en cuestión, debido a que no se encuentra ningún tipo de elemento que pudiera ser utilizado en la construcción y/o adecuación del proceso, así pues la definición de este proceso se hace desde cero, es decir, se define paso a paso cada uno de los elementos que son indispensables en la construcción de un modelo de procesos adecuado. Este será complementado a partir de la definición de nuevos elementos que permitan que el proceso ofrezca mejores productos software día tras día. Cada uno de los elementos que sean introducidos en el proceso serán debidamente estudiado y analizadas para su correcta adición, teniendo en cuenta las especificaciones dadas por el docente de la materia para que sean posibles y no se intervenga de manera errónea el proceso.

La metodología utilizada para este proyecto es Agil SPI Process, la cual describe un proceso de mejoramiento de procesos de software en 5 fases: Instalación del Programa, Diagnóstico, Formulación, Mejora, Revisión del Programa.

2.3 Entregables de Proyecto

FASE	ARTEFACTO	FECHA DE ENTREGA
Instalación del Programa	Lista de Riesgos	
	Carta de Gant	
Diagnóstico	Revisión del Proceso	
	Capacitación en TRP	
	Informe de Valoración del proceso	
	Lista de áreas de proceso priorizadas	
	Plan Técnico de implementación	
Formulación	Evaluación detallada de áreas prioritarias.	
	Modelos de proceso de las áreas de proceso prioritarias.	
	Manual de entrenamiento del proceso software.	
Mejora	Plan de Mejora	
Revisión del Programa	Lecciones Aprendidas	

Tabla 1. Entregables del Proyecto

2.4 Evolución del Plan del Proyecto

Cada uno de los documentos mencionados anteriormente podrá ser actualizado en reuniones programadas del grupo de QA, y también en reuniones eventuales que consideren cambios en los documentos, de esta manera habrá evolución continua del proceso de mejora que se esta desarrollando.

Organización del Proyecto

2.5 Estructura Organizacional

2.5.1 Equipo de gestión del proyecto de implementación.

Es la máxima autoridad del proyecto y responsable del éxito del mismo. Sus funciones serán:

- Aprobar el establecimiento de un grupo de Procesos dentro del curso de Pregrado (LSD) y de los ETP.
- Aprobar y apoyar los miembros del ETP.
- Proveer orientación al trabajo del ETP
- Apoyar la implementación de las recomendaciones del ETP que sean aprobadas por el comité.
- Monitorear y coordinar (seguimiento y control) las actividades del ETP.

Estará compuesto por:

- Cliente (Docente
- Ingeniero de Procesos(Tesista)
- Asesor(Experto en Agile SPI)

El comité se reunirá dos veces a la semana para conocer el avance del proyecto y definir temas de proceso que requieran de aprobación superior. Se deberá nombrar un responsable del ETP en el curso (LSD) que participara de las reuniones de seguimiento junto al asesor externo.

Se realizará al menos una reunión semanal con el asesor externo en Agile SPI de manera que el avance del proyecto tenga una revisión adecuada semana tras semana, estas reuniones serán concertadas por la líder de TIP

(Tesista) y el asesor, estas reuniones serán de vital importancia para la evolución del proyecto y la evaluación concertada del mismo.

En el contexto académico, este equipo esta conformado por dos personas (Docente y Tesista) quienes abordaran los roles necesarios para llevar a cabo de manera exitosa el proyecto, el asesor externo será el soporte para la ejecución del mismo.

2.5.2 Equipo de Tecnología de Procesos (Tesista – Asesor Agile SPI)

Es el responsable técnico del proyecto, cumpliendo las funciones de Grupo de Ingeniería del Proceso de Software. Sus funciones serán:

- Aprobar las definiciones a incorporar al proceso de desarrollo
- Coordinar las actividades de implementación de procesos de software
- Modelar y documentar los procesos definidos o modificados.

Se reunirá 3 veces a la semana los días lunes, martes y jueves en la franja horaria 08:00 – 10:00 Horas.

En este equipo participará personal definido por el Comité Ejecutivo, en principio:

- Consultor (Asesor Agile SPI)
- Ingeniero de Procesos (Tesista)

En este comité deberá participar una persona del curso LSD (Docente) durante todo el proyecto.

3 Proceso de Administración

3.1 Estimaciones del Proyecto

A definirse a partir de la fase de formulación.

3.2 Plan del Proyecto

3.2.1 Plan del proyecto

Refiérase a Carta Gantt del Proyecto (última Versión).

4 ANEXOS

4.1 Referencia documentos complementarios

A continuación se presentan los documentos que complementan este plan de implantación.

4.1.1 Lista de Riesgos.

Este documento refleja la lista de riesgos previamente visualizados que posiblemente afecten en alguna instancia el desarrollo normal de este proyecto.

Archivo Asociado: Lista de Riesgos

4.1.2 Carta Gantt.

Este documento tiene por objetivo representar los periodos de tiempo en los cuales se desarrollará el proyecto de mejoramiento continuo para el área de QA

4.1.3 Lecciones Aprendidas.

Este documento representa los aspectos que hoy basados en la primera mejora de procesos deben ser considerados a evaluar para mejorar.

Archivo Asociado: Lecciones Aprendidas

ANEXO C

GUIA DE INSTALACIÓN Y CONFIGURACIÓN DE LOS EQUIPOS

TECNOLOGÍA RPC

HERRAMIENTAS REQUERIDAS

1. Sistema Operativo Linux
2. Editor de texto
3. Librerías de C
4. Compilador de interfaces
5. Compilador de código fuente
6. Administrador de fuentes
7. Servidor de nombres

INSTALACIÓN DE HERRAMIENTAS

1. Instale el sistema operativo Linux en la distribución Ubuntu 8.04 o superior.

2. Instale o revise si ya se encuentra instalada la herramienta **Gedit** que se usara como editor de textos en las prácticas. Comando de instalación $apt-get install nombre_paquete.

3. Diríjase al Gestor de paquetes Synaptic e instale las librerías de C l**ibc6**. Comando de instalación $apt-get install nombre_paquete.

Verifique que **rpcgen** y **gcc** estén configurados correctamente, digite en una consola de comandos lo siguiente:

a. **$rpcgen**

Respuesta exitosa:usage : rpcgen infile

Respuesta no exitosa: bash: rpcgen: no se encontró la orden

4. Instale o revise si se encuentra instalado el **paquete build essentials** administrador de fuentes. Verifique que **make** está instalado correctamente con el siguiente comando :

   $make

   Respuesta exitosa:

   Respuesta no exitosa: bash: portmapper: no se encontró la orden

5. Por último instale o revise si se encuentra instalado **portmapper** servidor de nombres.

   Verifique que **portmapper** esté configurado correctamente, digite en una consola de comandos lo siguiente:

   b. **$portmapper –h localhost**

   Respuesta exitosa:

   Respuesta no exitosa: bash: portmapper: no se encontró la orden

PROBAR LA CONFIGURACIÓN

Para está fase se contará con una aplicación demo, con la que se probará que el entorno esta completamente configurado y listo para que los estudiantes puedan realizar sus prácticas sin inconveniente alguno.

1. Buscar en la carpeta del demo el archivo de la interfaz **nombre_interfaz.x**
2. Abrir una ventana de comandos y colocar el comando **$rpcgen nombre_ interfaz.x**

   Mediante el compilador de interfaces se generan los siguientes archivos:

   nombre_ interfaz _clnt.c: Contiene la funcionalidad del client stub.

   nombre_ interfaz.h: En este archivo de cabecera se definen estructuras declaradas en la interface remota.

   nombre_ interfaz _svc.c: Contiene la funcionalidad del server stub.

3. Generar las plantillas del cliente y el servidor utilizando los siguientes comandos:

$rpcgen –Sc nombre_ interfaz.x >cliente.c

$rpcgen –Ss nombre_ interfaz.x >servidor.c

Si el proceso se realizó correctamente se generan los siguientes archivos:

Cliente.c Plantilla del cliente de la aplicación.

Sevidor.c Plantilla del servidor de la aplicación.

3. Generar el archivo **makefile** utilizando los siguientes comandos en una ventana

   Makefile para el Cliente**: $ rpcgen -Sm nombre_ interfaz.x > makeC**

   Makefile para el Servidor: **$ rpcgen -Sm nombre_ interfaz.x > makeS**

4. Configuración del **makefile** generado para el servidor y el cliente.

   a. Cliente

      Editar y modificar los siguientes parámetros en el archivo **makeC** en la carpeta cliente.

      CLIENT : Nombre_Ejecutable

      SERVER: Borrar valor (Debe quedar en blanco)

      SOURCES_CLNT.c = cliente.c

      SOURCES_CLNT.h = nombre_interfaz.h

   b. Servidor

      Editar y modificar los siguientes parámetros en el archivo **makeC** en la carpeta servidor.

      CLIENT : Borrar valor (Debe quedar en blanco)

      SERVER: Nombre_Ejecutable

      SOURCES_SVC.c = servidor.c

      SOURCES_SVC.h = nombre_interfaz.h

5. Compilar el código fuente del cliente y del servidor con los siguientes comandos

      a. Para compilar los fuentes del cliente

         $ make -f makeC

      b. Para compilar los fuentes del servidor

         $ make -f makeS

6. Ejecute el Servidor de la aplicación con el siguiente comando:

$ ./nombre_servidor

7. Ejecute el Cliente que requiere como parámetro de entrada el nombre de la máquina en este caso localhost.

8. $ ./nombre_cliente localhost

Si siguió los pasos de esta guía ha configurado correctamente su equipo para realizar las prácticas referentes a la tecnología RPC del curso.

ANEXO D

Práctica 1: PROCESO BÁSICO DE DESARROLLO CON Sun RPC .

Ejercicio 1. (Será realizado en la Sala de Computo)

Diseñar un programa, para las elecciones de la gobernación del Cauca,usando Sun RPC. El sistema debe permitir:

- A un votante ingresar el voto por un candidato. Los votantes no pueden repetir el derecho al voto, por lo tanto el votante debe ingresar código de votante.
- A los periodistas les permite consultar la votación de un candidato particular, la votación del candidato ganador. Estas operaciones serán controladas mediante un Menú:

Basándose en la interface que se muestra en la Figura 1, diseñar el programa que se describió anteriormente de tal manera que toda la información de los votantes este en el servidor.

Los códigos de los candidatos a la gobernación son:

Nombre	Código
Campo L	107
Zambrano S	109
López J	111
Castro F	113
Ortega T	115
Vernaza R	117

Los códigos de los periodistas están dentro del rango de :200-299

Los códigos de los votantes van de 300 a 1000.

La solución de este ejercicio debe ser comprimida en formato zip o rar y subida al sitio. El nombre del archivo comprimido debe seguir el siguiente formato rpc-p1_apellido1_apellido2.zip. Donde apellido1 corresponde al primer apellido de uno de los integrantes y apellido2 corresponde al primer apellido del segundo integrante del grupo.

5 Crear el IDL

De acuerdo con el enunciado el usuario debe recurrir a un servicio que le permita usar el sistema de elecciones. Por lo tanto, el proceso Servidor deberá publicar mediante una definición de interface el procedimiento que atenderá la solicitud del proceso cliente. Esto se logra mediante un archivo con extensión .x pues en Sun RPC se definen las interfaces usando al notación XDR(External Data Representation).

Para cumplir con los requerimientos del problema se recurre a la definición de cinco funciones:

```
program ELEC_ALCALDE_PROG {
        version ELEC_ALCALDE_VERSION {
                int validar_votante(int cod_candidato)=1;
                void votar(int cod_candidato) = 2;
                int resultado_candidato(int cod_candidato) = 3;
                int votacion_candidato_ganador() = 4;
                int codigo_candidato_ganador()= 5;
        } = 1;
} = 0x20000001;
```

Figura 1: Interface remota eleccion.x

M
e
c
i
a
n
te un editor de texto cree la definición de interface indicada en la Figura 1.

2. Procesamiento de la Interface

Utilice el compilador de interfaces 'rpcgen' para procesar la definición de interfaces creada en el punto anterior. Desde una consola introducir el siguiente comando:

```
$ rpcgen eleccion.x
```

Mediante el compilador de interfaces se pueden generar siguientes archivos:

Cliente.c: Contiene la plantilla para el Programa de Usuario.

Servidor.c: Contiene la plantilla para las Rutinas de Servicio.

eleccion_clnt.c: Contiene la funcionalidad del client stub.

eleccion.h: En este archivo de cabecera se definen estructuras declaradas en la interface remota.

eleccion_svc.c: Contiene la funcionalidad del server stub.

- **Crear el código del Cliente.**

Sun RPC no posee un servicio binding de red global. En lugar de esto proporciona un servicio binding local llamado *portmapper*. Cada instancia del *portmapper* registra el puerto en uso por cada servicio ejecutándose localmente. Por lo tanto para importar una interface, el cliente debe especificar el *nombre del servidor*, *número del programa*, y *el número de versión*. El procedimiento *clnt_create()* es usado para este propósito. Este retorna un descriptor el cual es utilizado cuando un procedimiento remoto es llamado. El descriptor contiene la información necesaria para comunicarse con el puerto del servidor, tal como el identificador del socket y la dirección del socket.

El procesador de interfaces rpcgen permite generar templates(plantillas) del cliente y el servidor. Para generar el código del cliente vamos a utilizar esta facilidad.

Pasos a seguir:

a. Crear un subdirectorio denominado 'cliente'.

b. Mover/Copiar los archivos que estén relacionados con el soporte del cliente.

c. Ubicados en el subdirectorio 'cliente', modificar el código generado (archivo nombre_plantillaCliente, introduciendo la lógica de control del juego, y los mensajes guía para orientar la usuario.

4. Crear el código del Servidor.

De acuerdo al requerimiento es responsabilidad del programador del servicio incluir la funcionalidad de las operaciones básicas de este problema.

Para implementar el código del servidor nos basaremos en la plantilla de las Rutinas de Servicio.

Pasos a seguir:

a. Crear un subdirectorio 'servidor'.

b. Mover/Copiar los archivos relacionados con el soporte del Servidor al subdirectorio 'servidor'.

c. Modificar la plantilla del servidor para introducir la funcionalidad de cada función de búsqueda.

Un bloque del código generado se muestra a continuación:

5. Compilar códigos

Pasos a seguir:

a. Generar el archivo makefile

Otra opción que posee la aplicación rpcgen es la generación del archivo makefile, con el propósito de facilitar la compilación de los archivos fuente. Esto se logra mediante el siguiente comando:

$ rpcgen -Sm eleccion.x > makeC

Generar un archivo makefile tanto para el servidor como para el cliente.

b. Configurar el archivo makefile generado tanto para el cliente como para el servidor.

c. Compilar los fuentes:

Para compilar los fuentes del cliente

$ make -f makeC

Para compilar los fuentes del servidor

```
$ make -f makeS
```

6. Ejecutar el cliente y el servidor

Cuando se compilan los archivos fuente se generan dos ejecutables, cuyo nombre depende de los parámetros fijados en la plantilla del archivo makefile, para este caso especifico son: cliente y servidor.

El programa Servidor, se puede ejecutar localmente o en otra máquina.

```
$ ./nombre_servidor
```

El programa Cliente requiere como parámetro de entrada el nombre de la máquina (o dirección IP) donde esta el Servidor.

```
$ ./nombre_cliente localhost
```

ANEXO E

Requerimiento de RPC.

Ejercicio (Para ser entregado en la fecha indicada)

Condiciones de entrega:

El informe debe contener:

Manual de Usuario: Este documento servirá como referencia para el manejo de la aplicación.

Manual Técnico: Este documento contendrá el análisis y diseño de la aplicación, este documento servirá como referencia para el desarrollador de la aplicación.

Manual de Instalación: Documento donde se describe el proceso de instalación, por ejemplo estructuras de directorios, archivos de soporte, ubicación de los archivos fuente. Descripción del proceso de compilación.

Códigos fuente: Entrega de los códigos fuente debidamente documentados.

Fecha de entrega del requerimiento: 2 de Octubre de 2010.

Fecha de recepción de informes: 8 de Octubre de 2010

No hay fecha de aplazamiento. Por cada día de retraso, con respecto a la fecha de recepción de informes, se rebajará 1.0 sobre nota final. El informe cuyo código fuente no compile tiene un valor de 0.

Requerimiento:

El Grupo IDIS del Departamento de Sistemas está necesitando un sistema para gestionar la información de los proyectos y trabajos de grado, que se están desarrollando. El Sistema Distribuido debe estar basado en Sun RPC. El sistema posee tres tipos de usuarios un Administrador, estudiantes de fin de carrera, Investigadores(Docentes de Planta miembros del Grupo IDIS)

Funciones del Administrador:

El Administrador del sistema es el único usuario, con los privilegios para registrar a otros usuarios en el sistema. Sólo debe existir un sólo administrador en el sistema, por cada usuario registrado se requiere un login (con mínimo 8 caracteres) y un password(con mínimo 6 caracteres). El Administrador también esta autorizado para modificar o borrar una cuenta de usuario.

Funciones del Investigador:

El Investigador es el role asumido por los docentes de planta, que son miembros del Grupo IDIS. Las operaciones que pueden realizar estos usuarios son:

1 **Registrar la información de los proyectos de investigación y los proyectos de grado.**

2 Modificar cualquier tipo de información registrada en un proyecto de investigación y proyecto de grado.
3 Buscar un proyecto de grado o un proyecto de grado, usando como campo de búsqueda el código VRI (Entero comprendido entre 1-1000).
4 Borrar el registro de un proyecto o un proyecto de grado, usando como campo de búsqueda el código VRI (Entero comprendido entre 1-1000).
5 Imprimir en pantalla la bitácora de proyectos existentes, ya sea de proyectos de investigación o proyectos de grado.

Funciones del Estudiante:

El Estudiante es el role asumido por los estudiantes de sistemas de último semestre, que están habilitados para realizar un trabajo de grado. Las operaciones que pueden realizar estos usuarios son:

1.Buscar un proyecto de grado o un proyecto de grado, usando como campo de búsqueda el código VRI (Entero comprendido entre 1-1000).

2.Imprimir en pantalla la bitácora de proyectos existentes, ya sea de proyectos de investigación o proyectos de grado

Para tener acceso al sistema distribuido, los usuarios deben ingresar un login y un password.

Información que gestiona el sistema.

Información de usuarios:

Estudiantes

Nombre	Máximo 20 caracteres
Semestre	Semestre donde el estudiante está matriculado.
Login	Mínimo 8 caracteres
Password	Mínimo 6 caracteres

Investigadores

Nombre	Máximo 20 caracteres
Oficina	Lugar de trabajo del docente.
Login	Mínimo 8 caracteres
Password	Mínimo 6 caracteres

Administrador:

El login y el password, son definidos por el desarrollador del programa.

Proyecto de Investigación

Código VRI	Corresponde al número de registro en la VRI. Valor comprendido en el rango 1-1000
Título del Proyecto	Compuesto, máximo por 30 caracteres
Director	Nombre del docente responsable del proyecto, compuesto máximo por 25 caracteres.
Fecha de inicio	Formato seguido dd/mm/aa
Fecha de final	Formato seguido dd/mm/aa
Patrocinio	Compuesto máximo por 20 caracteres.

Proyecto de Trabajo de grado

Código VRI	Corresponde al número de registro en la VRI. Valor comprendido en el rango 1-1000
Título del Proyecto	Compuesto, máximo por 30 caracteres
Director	Nombre del docente responsable del proyecto, compuesto máximo por 25 caracteres.
Nombre de estudiantes	Compuesto máximo por 40 caracteres.
Fecha de inicio	Formato seguido dd/mm/aa
Fecha de final	Formato seguido dd/mm/aa

La información relacionada con los diferentes tipos de usuarios del sistema y la información de los proyectos de investigación y proyectos de trabajo de grado, deben ser almacenados en archivos de texto.

ANEXO F

Sustentación Requerimiento RPC

Cuestionario turno1.

Nombre:_____ Código:_____

Tenga en cuenta que:

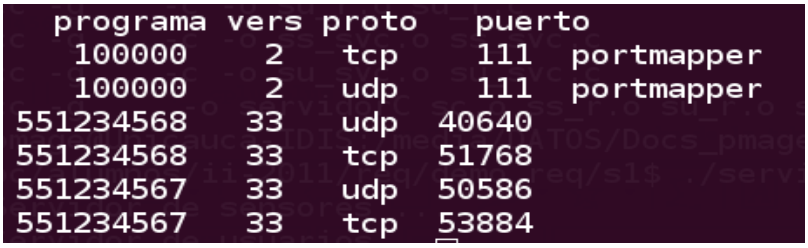
6 En cada respuesta debe incluir:
nombre del archivo: sentencia utilizada.

Para los comandos incluya:

nombre de host(role): comando

- El desarrollo de las respuestas se debe realizar entre 2 máquinas(1 Cliente , 1 Servidor).

1) Realizar los cambios que ud crea pertinente realizar en su aplicación, de tal manera que en el N_S se muestre la siguiente información, una vez ejecutada su aplicación:



Incluya las imagenes de pantalla donde evidencie los cambios que realizó en los archivos fuente y los comandos utilizados, para obtener dicho resultado.

2) Escoja una operación remota e incluya la secuencia de llamados operaciones que se hace desde que se invoca la operación remota, en el cliente, hasta que se hace el llamado de la operación en el servidor.

3) En cualquiera de los servidores incluir una operación (no remota) que permita contar las invocaciones realizadas a cualquiera de las operaciones de un servicio. El contador debe quedar almacenado en un archivo donde se muestre el siguiente contenido:

==========

Servidor X:

Invocaciones realizadas: 4

==========

Incluya las imagenes de pantalla donde evidencie los cambios que realizó en los archivos fuente, para obtener dicho resultado.

En la respuesta incluya:

nombre del archivo: sentencia utilizada.

Para los comandos incluya:

nombre de host(role): comando

En este archivo incluya sus respuestas, comprima este archivo, usando como nombre el siguiente formato: **req_sus_ape1.rar o zip** y subirlo a la plataforma usando el enlace del turno que le corresponde.