Universidad de las Ciencias Informáticas
Facultad 4

Título: Implementación de técnicas matemáticas de la Arquitectura de Información.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor: Ondrey Caballero Díaz.

Tutor: Mabel Medina Rodríguez.

Ciudad de la Habana, Julio, 2008
"Año 50 de la Revolución".

“Las matemáticas tienen invenciones muy sutiles y que pueden servir de mucho, tanto para contentar a los curiosos como para facilitar todas las artes y disminuir el trabajo de los hombres.”

René Descartes

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Ondrey Caballero Díaz

_____

Mabel Medina Rodríguez

_____

DATOS DE CONTACTO

Ondrey Caballero Díaz. Estudiante de la Facultad No. 4 de la Universidad de las Ciencias Informática.
Correo electrónico: ocaballero@estudiantes.uci.cu

Mabel Medina Rodríguez. Ingeniera en ciencias informáticas. Área de trabajo: Dirección técnica de la infraestructura productiva de la Universidad de las Ciencias Informáticas.
Correo electrónico: mmedina@uci.cu

Agradecimientos

A mis padres por haber confiado en mí durante todos estos Años.

A mi tío Roberto por preocuparse siempre por mí.

A mis abuelos y a mi familia en general por todo el apoyo que me han brindado.

A Rainier por estar siempre dispuesto a ayudarme.

A mis amigos(as) y compañeros(as) por hacerme pasar tan buenos momentos.

A todos los profes que contribuyeron de alguna manera en mi formación.

A Fidel, a Raúl y a la Revolución.

A todos, muchas gracias.

Dedicatoria

A mi mamá, y mi papá, por todos los valores que han sabido inculcarme.

A mis hermanas y a mis primos por apoyarme en todo momentos en mis decisiones.

A mi abuela Delia por rezar tanto por mí.

A mi tío Roberto por sus deseos de verme realizado como profesional.

A Sisley por mostrarme tantas cosas lindas en tan poco tiempo.

A todas las personas que me estiman, los que están… y los que ya no se encuentran a mi lado. Se que estrían muy orgullosos de mi.

Sientan todos que son parte de este trabajo. Una parte de mis resultados son suyos.

Gracias a todos por ser parte de mi vida.

RESUMEN

La arquitectura de información como disciplina en un proyecto Web es considerada de gran importancia pues su objetivo es obtener un resultado que se adapte a las necesidades y expectativas de la audiencia que utilizará el producto y esto es precisamente lo que se espera al desarrollar un software, la satisfacción plena del cliente. Para lograr este fin se han creado numerosas técnicas que forman parte y complementan el proceso de realización de una arquitectura de información en el ámbito de un proyecto Web. Algunas son fácilmente aplicables, otras, debido a la complejidad de su algoritmo, dificultan el correcto análisis de la información recopilada. Estas últimas son precisamente el objeto de estudio del presente trabajo de tesis, las técnicas matemáticas de estructuración, agrupación y organización de contenidos, Card Sorting y Análisis de secuencia. Partiendo de la investigación anterior, se propone como resultado de este trabajo la implementación de las funcionalidades de dichas técnicas para obtener una aplicación que simplifique y perfeccione el trabajo de los arquitectos de información a la hora de realizar una arquitectura de información usable, accesible y centrada en el usuario.

Palabras claves:
Arquitectura de información, audiencia, software, proyecto Web, Card Sorting, Análisis de Secuencia, usable, accesible.

TABLA DE CONTENIDO

INTRODUCCIÓN

En la actualidad cubana, la Web se ha convertido en un recurso informático muy importante para todos los aspectos de la vida, la salud, educación, cultura, deportes, economía, política, entretenimiento, entre otros. Muchos de los problemas que vemos cuando estamos realizando alguna búsqueda en Internet o revisando algún sitio Web en especial, son originados cuando las empresas ponen toda su atención en la tecnología y descuidan la dimensión humana de sus sistemas; La tecnología es fundamental, pero insuficiente, ya que de nada sirve tener un sitio o multimedia vanguardia si nuestros clientes o usuarios tienen problemas para usarlo. Con el propósito de resolver estos problemas surge la Arquitectura de Información.

La Arquitectura de Información *"organiza patrones inherentes a la información, haciendo entendible lo complejo"* o también se ve *"como un mapa de información que permite a otros encontrar su vía personal hacia el conocimiento"* (*Wurman, 1975*), es una disciplina, surgida ante la necesidad de organizar conjuntos de información, permitiendo que cualquier persona los entienda y los integre a su propio conocimiento de la manera más simple posible. Esta arquitectura se usa fundamentalmente en espacios virtuales como los sitios Web de Internet, donde se requiere que el propio usuario obtenga la información, sin ayuda de terceros. Comprende además los sistemas de organización y estructuración de los contenidos, los sistemas de rotulado o etiquetado de dichos contenidos, y los sistemas de recuperación de información y navegación que provea el sitio Web.

Los encargados de poner en práctica este tipo de arquitectura, son los arquitectos de Información, los cuales han estado utilizando diferentes métodos para diseñar sitios Web y multimedia de mayor calidad guiándose primeramente por su criterio para establecer el orden y diseño de cómo va la información en dichos sitios y luego utilizando algunas técnicas brindadas por la Arquitectura de Información que fueron surgiendo con dicho propósito, tales como las entrevistas con los usuarios, tormenta de ideas, etc. Aunque también están las técnicas de categorización de contenidos por medio de los diagramas de afinidad y la selección de tarjetas *(Card Sorting y Análisis de Secuencia)*, Aplicar estas técnicas permite alcanzar una mayor calidad, eficiencia y competitividad a la hora de realizar mejores diseños.

Gracias a la Arquitectura de Información, y las técnicas que brinda para la categorización de contenidos, se pueden obtener buenos resultados ante la realización de sitios Web y estos son

comprobados cuando los usuarios llegan a comprender el sistema y llegan a realizar la tarea por la cual se dirigieron al mismo.

Existen dos enfoques para la realización de sitios Web que permiten lo antes mencionado; estos son ***usabilidad y accesibilidad.*** Ambos mejoran la efectividad, eficiencia y satisfacción de los usuarios. Cuando alguna persona navega en la Web, es porque básicamente está en busca de algo, por esa razón, hacer sitios con las características antes mencionadas, para evitar que las personas se sientan extraviadas, perdidas o defraudadas se ha convertido en una necesidad para los diseñadores y arquitectos de interfaces.

Tenemos entonces que: "*Usabilidad es un atributo de calidad que mide lo fáciles de usar que son las interfaces Web*" (*Jakob Nielsen, 2003*), o sea la ***usabilidad*** abarca todo el comportamiento humano que se viene evidenciando cuando alguna persona desea realizar alguna tarea específica de la forma más sencilla y eficaz frente a un ordenador sobre algún entorno gráfico o Web. Por otra parte, la ***accesibilidad*** ya no se refiere a la facilidad de uso, sino a la posibilidad de acceso a la información. En concreto: *"que el diseño como prerrequisito imprescindible para ser usable, posibilite el acceso a todos sus potenciales usuarios, sin excluir a aquellos con limitaciones individuales: discapacidades, domino del idioma, limitaciones derivadas del contexto de acceso (software y hardware empleados para acceder a la información, así como el ancho de banda empleado)"* (*Hassan Montero, Martín Fernández*; 2003).

Una de las mayores problemáticas existentes en la actualidad y que trae consigo muchas quejas por parte de los usuarios, es que en muchos sitios Web existe mucha información desorganizada, donde pasan mucho trabajo buscando los datos que necesitan o simplemente se aburren de estar navegando sin encontrar en dicho sitio lo que necesitan; y esto hace que el sitio creado no sea ***usable*** o lo que es lo mismo, puede decirse que el acceso a ese sitio será muy pobre ya que los usuarios lo consideraran desde inseguro hasta como una pérdida de tiempo el recurrir al servicio que brinda, lo cual traerá un significado negativo tanto para la imagen del diseñador así como para la entidad laboral a la cual este representando, además de que la calidad del producto final no será la deseada y por tanto las costos requeridos para poner en funcionamiento el sitio y para su posterior mantenimiento aumentarán. Lo importante de la creación de un sitio es que se mantenga y sirva de buen uso para las personas necesitadas de la información que el mismo pueda brindar. En Internet, como en cualquier sistema de información, el ahorro de tiempo es una gran necesidad, por lo que mientras más nos preocupemos de conocer a nuestros usuarios y diseñar pensando en ellos, más frecuentes será el éxito.

Con el surgimiento de la Universidad de las Ciencias Informáticas (UCI), en los últimos años se ha evidenciado un creciente desarrollo tecnológico y con este, un marcado auge en el desarrollo de

aplicaciones Web y multimedia, aplicadas en las diferentes esferas de la sociedad. Por lo que los Ingenieros y diseñadores han comenzado a hacer uso de las técnicas antes mencionadas, pero de la forma en que se han venido desarrollando desde hace algunos años, realizando todo el trabajo manualmente. Es un hecho que el proceso de realización de pruebas y obtención de resultados que requieren *Card Sorting* y *Análisis de Secuencia* es muy engorroso y consume mucho tiempo. Por lo que se pretende que la realización de este trabajo contribuya a la obtención de una aplicación que ayude a explotar con mayor facilidad las oportunidades que nos brinda la Arquitectura de Información de diseñar y construir sistemas más amigables y eficientes.

Debemos señalar que las técnicas matemáticas de concurrencia de la Arquitectura de Información son unas de las más importantes a tener en cuenta a la hora de realizar la categorización y ubicación de contenidos de un sitio Web. Estas técnicas matemáticas son de tipo encuesta, y dado que hay que realizárselas a un número aleatorio de usuarios potenciales del producto final, la obtención de los resultados es un poco complicada y requiere incluso en ocasiones un trabajo en equipo de los arquitectos para obtener los resultados más óptimos para realizar la agrupación, estructuración y organización de los contenidos de cualquier sistema que brinde información. Por otra parte, la obtención de resultados erróneos luego de haber aplicado las técnicas *Card Sorting y Análisis de Secuencia* ha provocado diseños de poca calidad lo cual conlleva a gran insatisfacción por parte de los clientes y usuarios potenciales de los productos. No aplicar estas técnicas es considerado un error que afecta la *navegabilidad, usabilidad y accesibilidad* de nuestros sitios y una carencia de Diseño Centrado al Usuario (DCU).

Aprovechando el proceso de independización tecnológica que se viene desarrollando en al UCI, surge la idea de realizar la implementación de una aplicación que modele los resultados de estas técnicas, ante la necesidad de optimizar y acelerar el trabajo de los diseñadores Web y Arquitectos de Información de la Universidad. Esta aplicación debe controlar todo el proceso referente al funcionamiento y devolución de resultados de las técnicas de la Arquitectura de Información Card Sorting y Análisis de Secuencia las cuales serán los temas fundamentales que se trataran en este trabajo.

El problema principal consiste en ¿Cómo desarrollar una herramienta que automatice las funcionalidades brindadas por las Técnicas Matemáticas de la Arquitectura de Información *Card Sorting y Análisis de Secuencia* para realizar el proceso agrupación, organización y estructuración de los contenidos de cualquier sistema que brinde información?

El **objeto de estudio** del presente trabajo son las técnicas matemáticas Card Sorting y Análisis de Secuencia.

Para responder al problema científico anteriormente planteado nos trazamos como **objetivo general**: Implementación de una aplicación, que permita automatizar las funcionalidades de técnicas matemáticas de la Arquitectura de Información para agilizar y optimizar la toma de decisiones por parte de los diseñadores y Arquitectos de Información en la Universidad de las Ciencias Informáticas.

Para dar cumplimiento al objetivo general planteado, se trazaron algunos objetivos específicos, los cuales se enumeran a continuación:

- Definir cómo se realizan las técnicas de tipo encuesta, en especial Card Sorting y Análisis de Secuencia.
- Documentar sobre los diferentes sistemas y software donde se utilizan técnicas de categorización de contenidos por medio de selección de tarjetas que existen actualmente en el resto del mundo.
- Modelar y diseñar una solución para implementar las funcionalidades de Card Sorting y Análisis de Secuencia
- Implementar la aplicación.

Para responder a los objetivos específicos llevaremos a cabo las siguientes **tareas**:

- Investigar como se desarrollan las diferentes técnicas matemáticas propuestas.
- Revisión bibliográfica sobre los diferentes métodos y aplicaciones existentes en la actualidad en el mundo para evaluar, identificar y proponer una solución al problema que da paso a esta investigación.
- Estudiar las etapas de desarrollo del software basado en alguna de las metodologías de desarrollo que existen actualmente.
- Estudiar el lenguaje de programación en el cual se implementará.
- Analizar y definir las tecnologías a utilizar para el desarrollo de la aplicación.

Del objeto de estudio derivamos que el **campo de acción** que abarca este trabajo, es el estudio de Técnicas y Modelos Matemáticos.

Como **aportes prácticos** esperados con el desarrollo del trabajo tenemos que se logrará disminuir la tarea de los diseñadores y analistas de sitios Web de la Universidad de las Ciencias Informáticas a la hora de realizar la agrupación, organización y estructuración de la información. También ganarán

tiempo a la hora de realizar los análisis gráficos y modelados de los resultados. Por último, se espera lograr que los usuarios obtengan un medio de trabajo más cómodo, agradable, amigable y eficiente posible.

Las tareas de la investigación llevan a una estructuración del contenido distribuida en cuatro capítulos, como se especifica a continuación.

Capitulo 1: El propósito de este capítulo es la formalización de todos los conceptos asociados al tema y que son necesarios para el mejor entendimiento de lo que se describe en el resto del trabajo. Se analizarán algunas soluciones antes desarrollas, así como sus ventajas, desventajas y si solucionaban de alguna manera los problemas de los usuarios. También se analizará la tecnología y metodologías a utilizarse.

Capitulo 2: El propósito de este capítulo es la descripción del objeto de estudio del presente trabajo. Se describirá la propuesta del sistema como solución a los problemas que dan paso a esta investigación, así como la definición de los requisitos no funcionales como funcionales, de este último se definirán los casos de usos, detallando aquellos de gran significado en el desarrollo de la solución.

Capitulo 3: Se realizara el análisis y diseño detallado de la solución planteada en el capítulo 2

Capitulo 4: El objetivo de este capítulo es realizar la solución propuesta e implementada, así como realizar las pruebas de la misma y explicar cómo funciona.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

Con la evolución constante de la tecnología y la creación de incontables dispositivos y aplicaciones que automatizan muchas de las tareas en los diferentes sectores de la sociedad, se ha evidenciado un incremento apreciable en la productividad de las empresas, en la rapidez de creación y lanzamiento de los productos así como mejoras considerables en la calidad. Estos aportes prácticos ayudan considerablemente a mejorar y facilitar la labor y las tareas de muchos trabajadores de incontables empresas.

En la Universidad de las Ciencias Informáticas casi toda la información fluye por medio de aplicaciones Web, multimedia, y foros de discusión, además de que se realizan diferentes actividades con objetivos educativos deportivos y políticos, como son por ejemplo los eventos "Mi Web por Cuba", evento "Juvenil Martiano" en los cuales se crean muchos sitios Web hablando sobre variados temas. Estas razones son las que hacen importante el uso de la *Arquitectura de Información* ya que su uso permite optimizar la calidad a la hora de crear sitios Web y multimedia para que sean más accesible y usables y así proporcionar a todas aquellas personas con algún tipo de discapacidad, una oportunidad de tener un acceso equitativo a la información e igualdad de oportunidades a la hora de interactuar con la sociedad.

En este primer capítulo se analizarán diferentes aplicaciones que se han creado para mejorar la Arquitectura de Información, así como trataremos sobre las diferentes técnicas usadas en esta arquitectura. También se explicará la metodología, diferentes herramientas de desarrollo y tecnología de apoyo para resolver el problema planteado.

1.2 Técnicas de la Arquitectura de Información.

En las dos últimas décadas, la Arquitectura de Información se ha ganado un lugar en los planes de estudio de las carreras relacionadas con las Ciencias de la Información. Aunque es una disciplina que está siendo conocida y explotada en este período, su definición surge a partir de 1975, ante el crecimiento exponencial de la información y los recursos informativos en el mundo y la necesidad de gestionarlos. Constituye los procesos de planificación y organización de la información dentro del ciclo de vida de algún producto electrónico y ha ganado su mayor popularidad con el desarrollo de sitios Web en Internet e Intranets. La Arquitectura de Información, también fue definida por Richard Saul Wurman *como "una combinación de la organización de la información del contenido del sitio en categorías y la creación de una interfaz para sostener esas categorías" (Gómez Reyes; 2002).* Esta Ar*quitectura* cuenta con algunos procesos para lograr esta organización por categorías; entre las más importantes se encuentran:

- Obtener la visión y objetivos del mismo, definiendo las necesidades del diseñador y las de a los que va dirigido la aplicación.
- Definir el tipo de funcionalidad que el sitio va a tener y además, la información que el mismo va a tener.
- Presentar la evolución posible que podría tener el sitio en un futuro, como se adaptará a los cambios y posible crecimiento que este podría tener en el paso del tiempo.

Para lograr estos procesos de organización, estructuración, representación y diseño centrado al usuario, la *Arquitectura de Información* ha definido algunas técnicas para aportarle mayor calidad a los productos electrónicos desarrollados. Estas son:

1.2.1 Técnicas de interacción con el usuario:

Se realiza por medio de entrevistas, encuestas, realización del diseño de escenarios, así como el diseño participativo. El objetivo de este tipo de técnica es la de obtener información relacionada con el usuario del producto final, es considerada como la base para lograr el diseño centrado en el usuario para sustentar las etapas de producción del producto. Esta técnica se realiza mediante el contacto directo con las personas a quienes irá dirigido el producto final y existen varias formas de aplicarla entre las que se encuentran:

Las ***reuniones***, las cuales se realizaran en todas las etapas de producción del producto. Otra forma es por medio de las ***entrevistas y encuestas*** donde se contactan a los usuarios personalmente para de esa manera obtener información que posteriormente pueda ser analizada. También se encuentra el ***diseño de escenarios*** donde se les pide a los usuarios definir el orden de las acciones, para obtener directamente de ellos las secuencias lógicas para el desarrollo del producto. Por ultimo se encuentra el ***diseño participativo*** donde se realiza una reunión con usuarios potenciales junto con los diseñadores para la realización del diseño del producto.

1.2.2 Técnicas de interacción con el contexto:

Esta técnica se basa en el estudio y ***evaluación de productos similares***, se desarrollará una vez trazados los objetivos de nuestro producto, para así buscar similitudes en otros productos que tengan semejanzas al nuestro para luego definir indicadores que serán evaluados y llegar a resultados sobre cual es la mejor manera usar los mismos. Y el segundo caso es el ***análisis de la competencia*** la cual no solo comprende el análisis de otros productos, sino las instituciones que lo desarrollan, conocer los errores que cometieron, para no incurrir en ellos, ahorrar tiempo y ganar calidad en el trabajo desarrollado.

1.2.3 Técnicas matemáticas (coocurrencia):

Con la aplicación de estas técnicas se crean grupos aplicando la co-ocurrencia, y formando grupos permitiendo crear secuencias correspondientes con el modelo mental de los usuarios, haciendo mas

fácil y precisa la toma de decisiones. Como es sabido y se han tratado ya estos términos; dentro de esta técnica se encuentran la ***organización de tarjetas (Card Sorting)*** y el ***Análisis de Secuencia.***
En *Card Sorting* se crean un grupo de tarjetas que cada una contenga cada termino que fueron siendo obtenidos por parte de los usuarios según sus necesidades y los términos surgidos mediante el estudio del contexto. Estas tarjetas se les entregan a los usuarios pidiéndole que las organicen según su criterio, y mientras hacen esto se va midiendo y evaluando cuales de los grupos son los de mayor complejidad, dudas del usuario, etc. Luego se analizan los resultados mediante la obtención de una tabla de coocurrencia y un dendograma interactivo. Por otro lado tenemos que el objetivo del Análisis del producto, como por ejemplo el establecimiento de la secuencia de términos en una barra de navegación, menú desplegable o un listado de productos que serán vendidos, etc.

1.2.4 Técnicas de representación de información:

Este tipo de técnicas se utilizan después de haber obtenido toda la información que brindaron las técnicas anteriores antes descritas. Ayudan a concretar modelos de lo que sería el producto final mediante la creación de prototipos y a retroalimentación entre diseñadores y usuario.
Dentro de estas técnicas se encuentran la ***diagramación y bocetado*** que permite, a través de diagramas concretar la propuesta de los diseños realizados por los arquitectos de información. Otra es la ***representación de etiquetas,*** es que se basa en ponerle etiquetas a todos los textos que se usarán en los títulos, hipervínculos, etc. Por último tenemos el ***prototipado y maqueta,*** que muestra el producto sin acabado final, son maquetas que estarán en relación directa con los diagramas realizados y la representación de etiquetas utilizadas.
La utilización de estas técnicas nos confirma la necesidad existente de organización de la información para lograr productos electrónicos más usables. Estas técnicas están interrelacionadas entre si y se nutren mutuamente de información; la primera a la segunda, y esta a su vez facilita información a la tercera y esta nutra a la siguiente, lográndose una concatenación entre todas. Por tanto podemos afirmar que la utilización de estas técnicas ayuda al proceso de creación de productos de mayor calidad y mayor satisfacción del producto final.

1.3 Software y aplicaciones desarrolladas internacionalmente que automatizan las técnicas de la Arquitectura de Información.

En la Sección anterior se hizo alusión a las técnicas de la Arquitectura de Información y los procesos de organización, estructuración y representación de la información, con el objetivo de apreciar su interrelación y concatenación. En esta sección nos enfocaremos en especial sobre las ***técnicas matemáticas*** y los diferentes software desarrollados donde se implementen de alguna manera el funcionamiento y comportamiento de las técnicas de categorización de contenidos Card Sorting y Análisis de Secuencia. Algunos de estas aplicaciones se asemejan y otras difieren en algunos

aspectos en su funcionamiento, pero tienen el mismo objetivo de facilitar la tarea de los Arquitectos y diseñadores para así contribuir a favor del auge de la Arquitectura de Información. Dentro de los softwares que se han creado y utilizado se encuentran: WebSort, EzSort, CardSort, CardZort.

1.3.1 WebSort:

Es una herramienta online que es utilizada para realizar los estudios de sorteo de tarjetas. Esta herramienta cuenta con la llamada Interfaz del Gerente, la cual es utilizada para configurar el estudio que se pretende levar a cabo. Una vez creado el estudio se pueden poner en el panel todos los artículos a los cuales se les hará la selección y a su vez, es opcional el realizar una breve descripción de los mismos. En el caso de realizar el Card Sorting de tipo *cerrado,* el realizador del estudio predefinirá los grupos entre los cuales serán repartidos los artículos, y en caso de ser *abierto* los usuarios podrá crear los grupos a conveniencia, modificarlos, cambiar nombres de los mismos, etc.

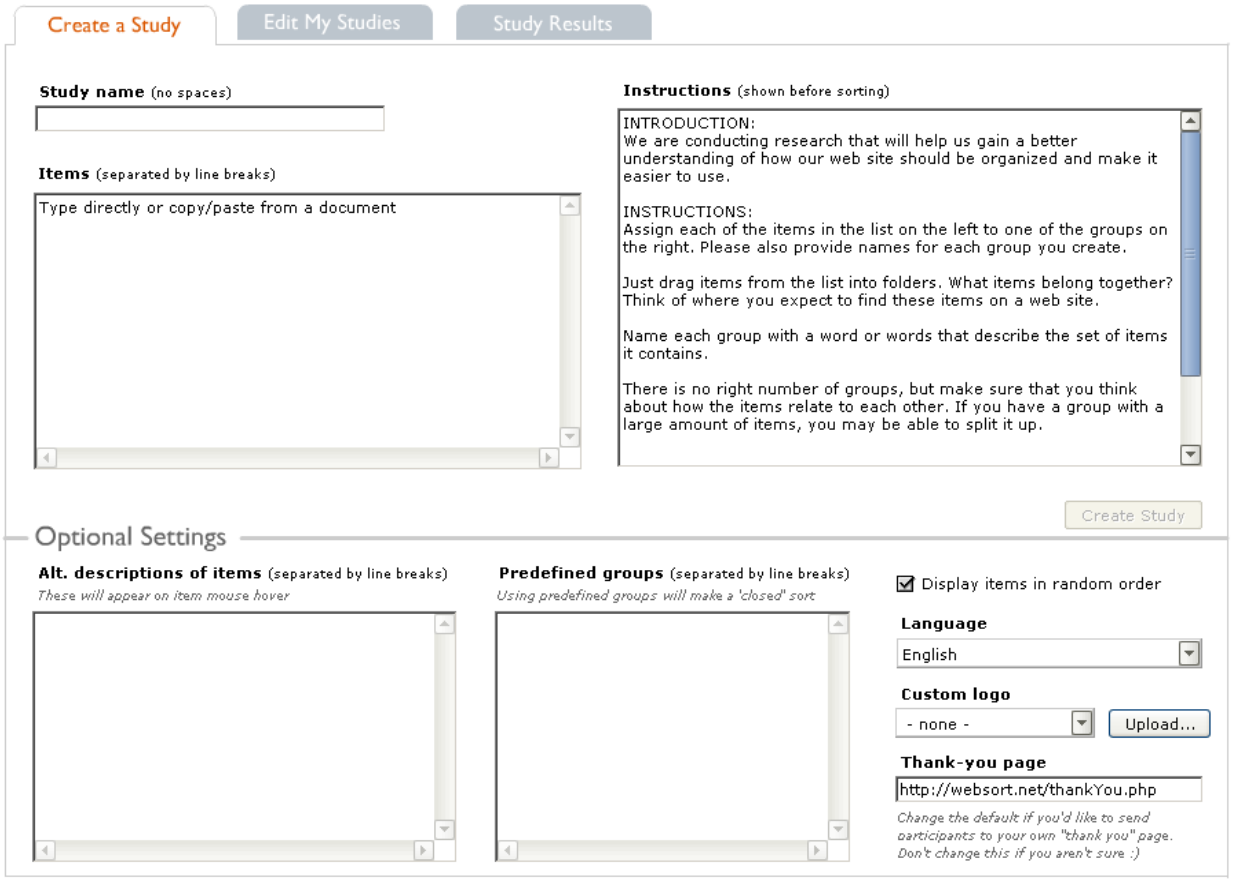


Figura 1.3.1-1 Interfaz del Gerente. (Para configurar estudio)

Tiene una interfaz amigable y se hace sencillo realizar el sorteo de las tarjetas y su modificación. Cada estudio realizado por los participantes es guardado en la base de datos del WebSort, y estos datos son accedidos por el creador de la prueba luego de ser avisado de que el último participante ha completado su estudio entonces se pueden ver las categorías individuales y los artículos que ha

colocado en cada una de esas categorías, junto a la lista generada por la aplicación de todos los participantes que estuvieron realizando la prueba.

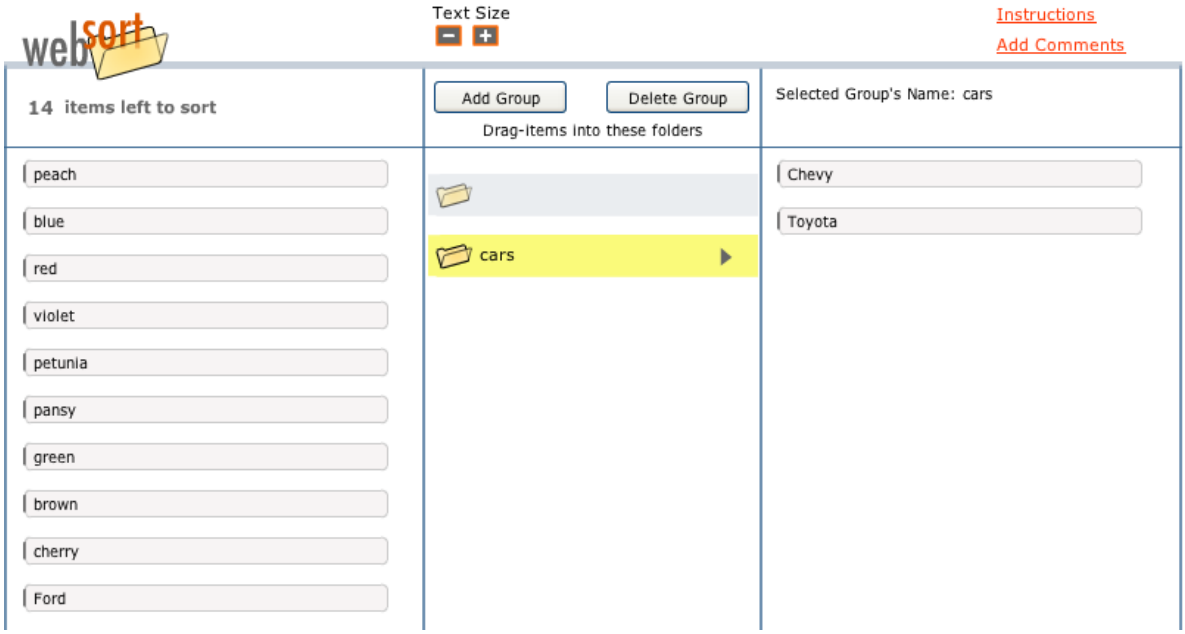


Figura 1.3.1-2 Artículos y grupos.

Otra forma de analizar los resultados generados por esta aplicación es mediante el análisis del *cluster,* que no es más que la creación de un diagrama en forma de árbol, el cual corresponde a un tipo de promedio de todas las agrupaciones hechas, ya sea la realizada individualmente por cada participante, como los promedios de todos juntos como se muestra en la siguiente imagen.

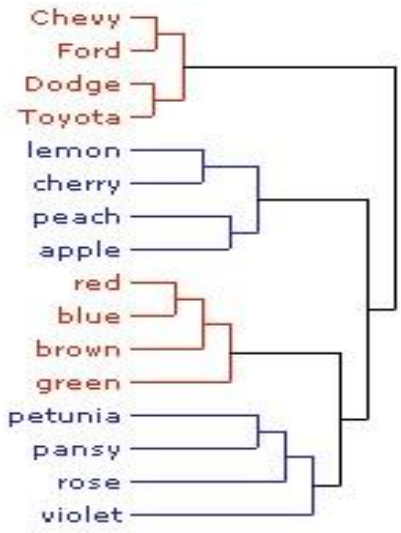


Figura 1.3.1-3 Análisis del cluster.

WebSort es una aplicación Web, que pertenece a la empresa Parallax creada por Larry y Jed Wood, en el estado de Utha, USA, quienes son los propietarios y diseñadores de dicho sitio, URL, así como de todo el contenido, software y material que se expone en el mismo. En la actualidad Parallax ofrece un descuento del 50 % a instituciones de caridad y educativas. Esta aplicación a pesar de ser muy

eficiente, solamente responde al funcionamiento de técnica de Card Sorting, dejando fuera de su alcance todo lo relacionado con el Análisis de Secuencia, además de ser un software propietario lo que implica tener que pagar una licencia para poder utilizarla. Por otra parte, el hecho de que la realización de esta prueba sea por medio de la red, donde el creador de la misma obtiene una dirección URL donde los usuarios se conectarán para realizar la prueba, hace que el termino *"interacción directa con el usuario"* carezca de sentido.

1.3.2 CardSort:

Esta es otra de las herramientas para mejorar la Arquitectura de la Información que utiliza la técnica de Card Sorting. Fue creada en el 2003 por Steffen Schilb durante la realización de su trabajo de diploma. Dicha aplicación presenta una interfaz sencilla de trabajo donde en un inicio se puede seleccionar el vínculo para realizar la creación y edición de las tarjetas y los artículos, así como crear los nombres de los grupos de las categorías a las cuales pertenecerán dichos artículos. Por otro lado, también se podrá realizar el sorteo de dicho artículos, creando los grupos según el usuario crea conveniente.

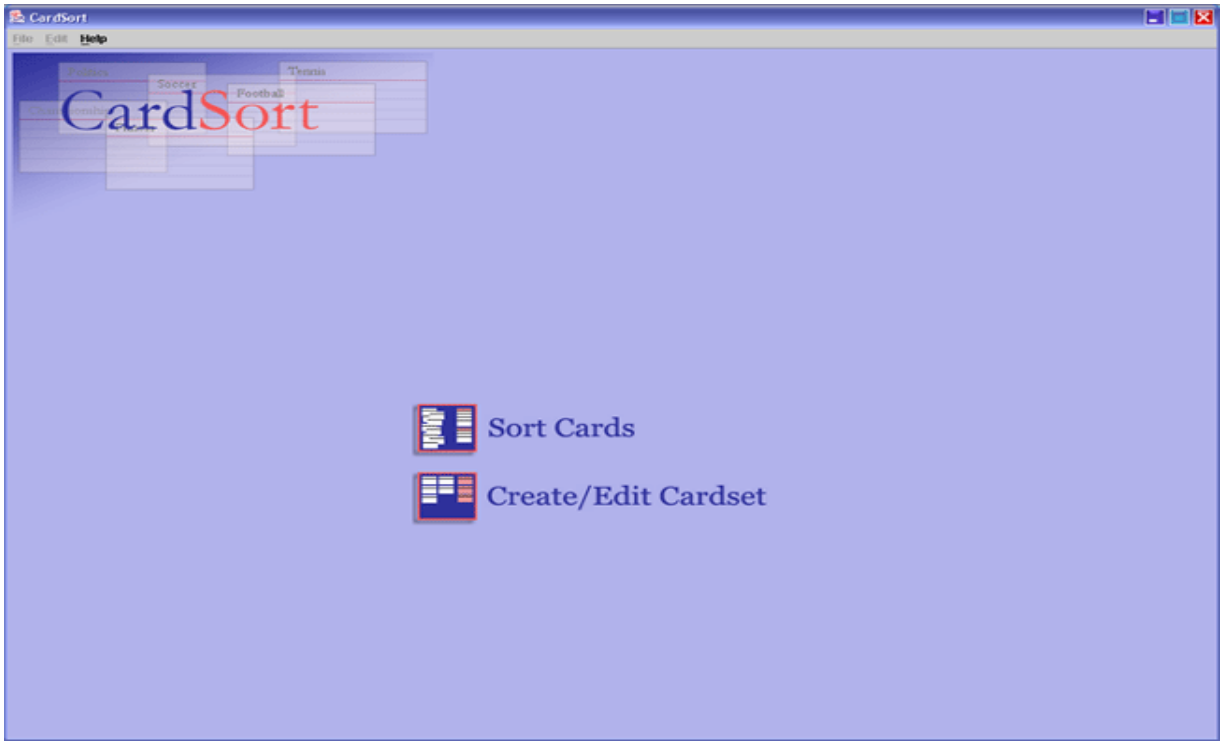


Figura 1.3.2-1 Inicio de CardSort

Otra de las ventajas de esta aplicación, es que permite crear un máximo de cien tarjetas para cada estudio lo cual es una cifra considerable dado que se podrán tener en cuenta gran cantidad de artículos a realizarle la selección.

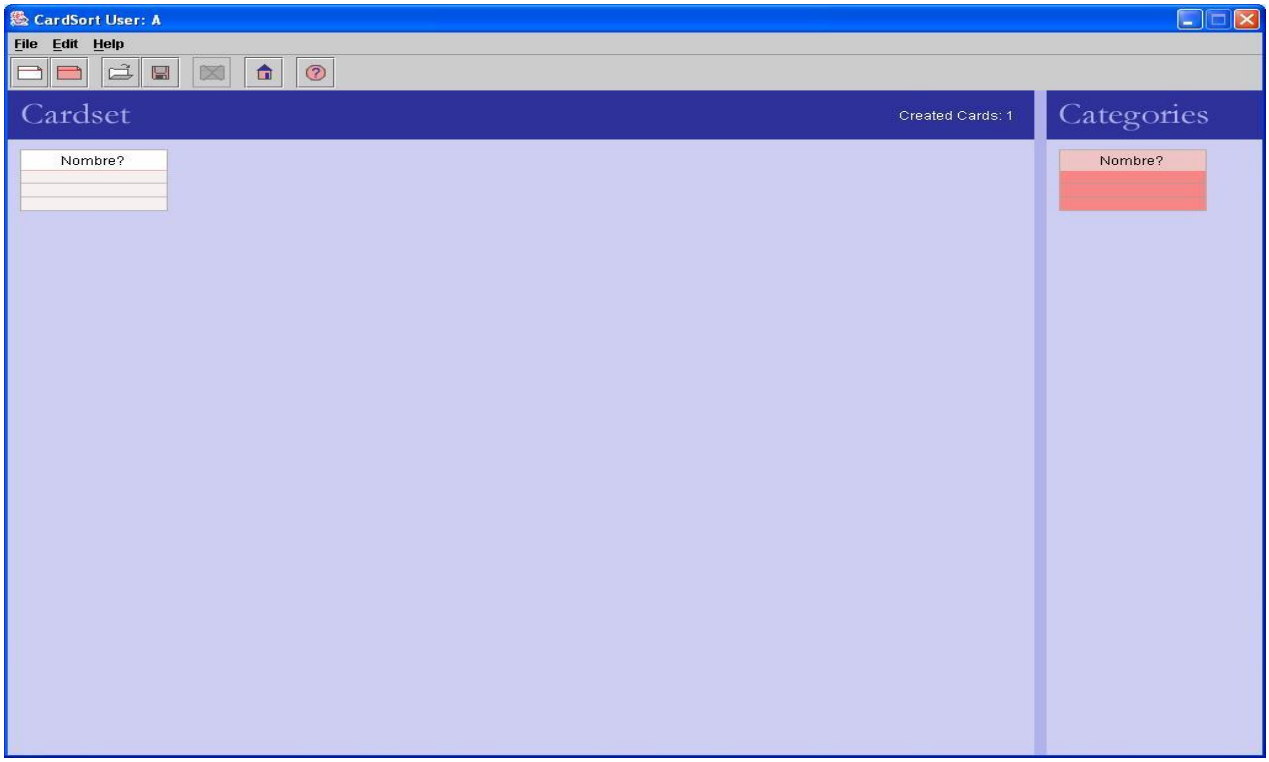


Figura 1.3.2-2 Creación de tarjetas y categorías.

Por otra parte, se podrá guardar un archivo con la creación de dicho grupo de tarjetas el cual será cargado posteriormente en la sesión de realización del sorteo, mostrando en el panel izquierdo el grupo de artículos, y en el derecho el grupo de categorías (en caso de que fueran creadas) en el cual se repartirán dichos artículos.

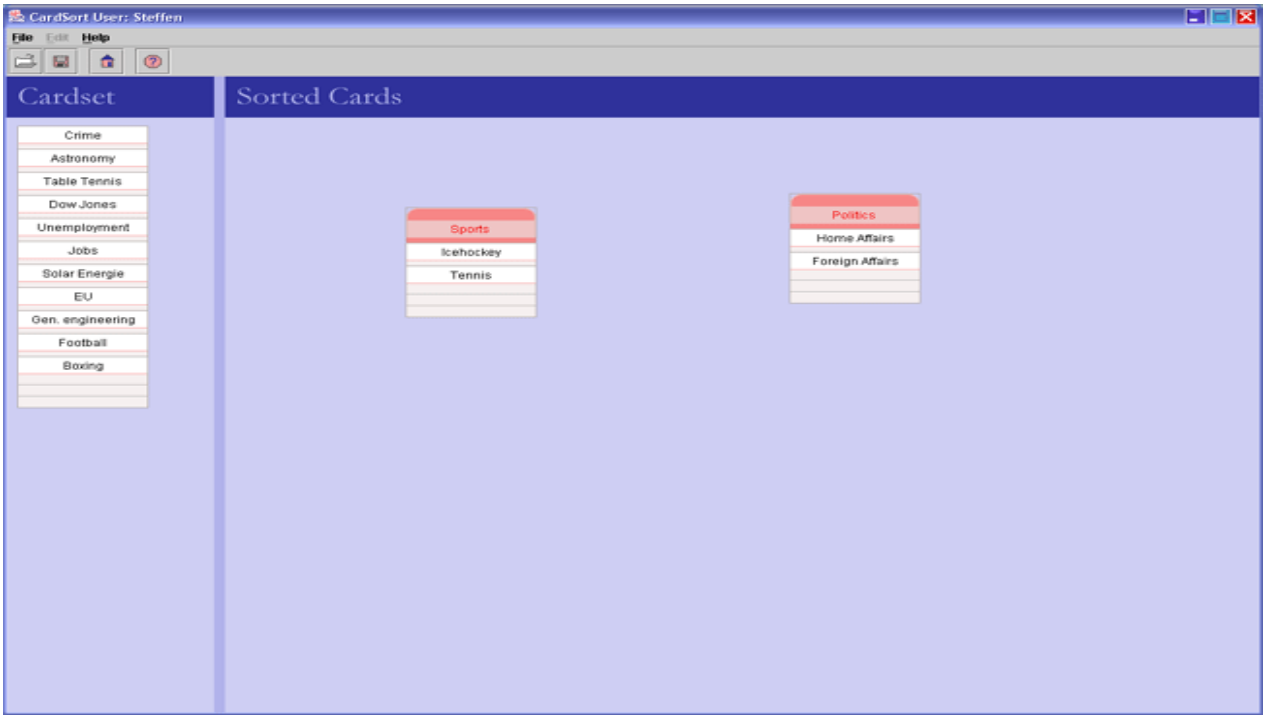


Figura 1.3.2-3 tarjetas(a la izquierda), Grupos o categorías (a la derecha)

Al final, quedará una selección de tarjetas separadas en diferentes grupos, y dicha selección se salvará por cada realizador de la prueba y se enviará al responsable de la misma, el cual se encargará de realizar la búsqueda de los resultados.

CardSort fue programada en java, bajo el ambiente de programación eclipse, y aunque es un software libre, sirve de bien poco pues esta herramienta solo puede ser utilizada para la creación de las tarjetas y grupos, así como para la realización del sorteo de las mismas, ya que para la realización y obtención de los resultados utiliza los software *EzCalc* y *EzSort,* los cuales fueron creados por IMB (*empresa creadora y comercializador de herramientas, programas y servicios relacionados con la informática*), pero que desde hace varios años no han sido actualizados y fueron retirados de sus respectivos sitios de descargas.

Los datos antes mencionados, constituyen una gran desventaja para CardSort lo cual lo convierte en un programa inutilizable para nuestros propósitos.

1.3.3 CardZort:

Esta es otra de las aplicaciones informáticas que permite la rápida creación y ejecución de ejercicios de selección de tarjetas y el análisis de los grupos resultantes. Puede ser ejecutada sobre la plataforma de Windows y posee una interfaz gráfica muy amigable que imita en muchos aspectos el verdadero proceso de selección de tarjetas. Las tarjetas son creadas donde se les pude poner un título y una descripción a cada una de ellas como se muestra en el ejemplo de la siguiente figura:

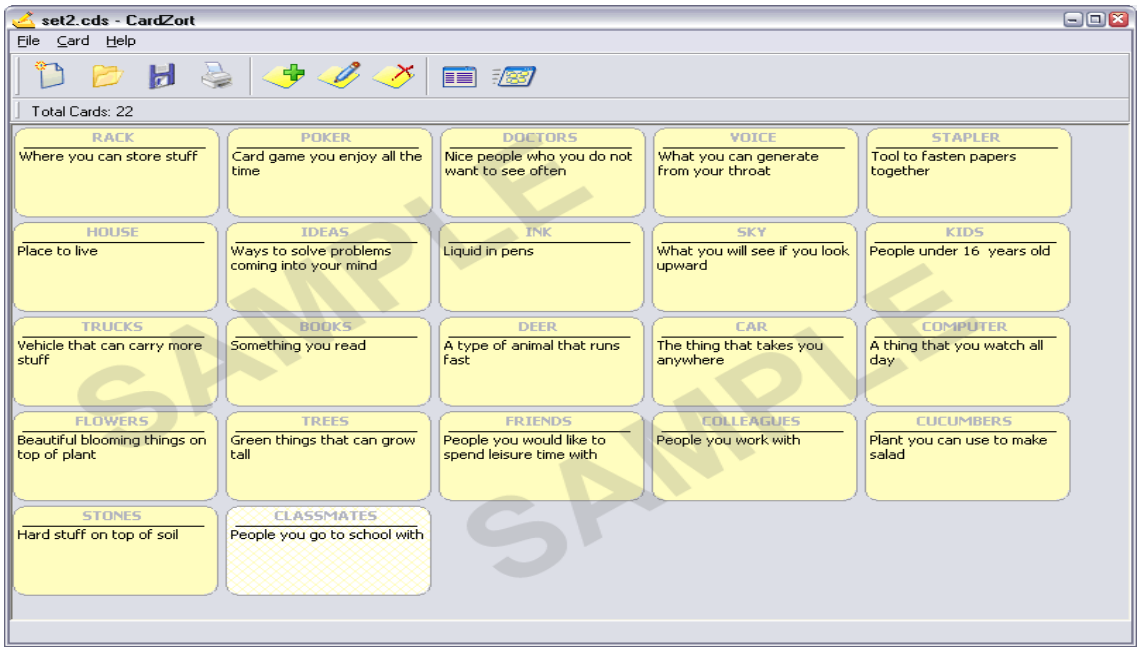


Figura 1.3.3-1 Creación de tarjetas.

Luego de creadas, estas se van agrupando en montículos, para así ir conformando los grupos de categorías que formaran parte de la selección final, y estos pasos son muy similares a como se hace a

mano pues se van arrastrando una sobre otras las tarjetas que conformarán cada uno de los grupos. (Esto es el primero de los pasos a realizarse).

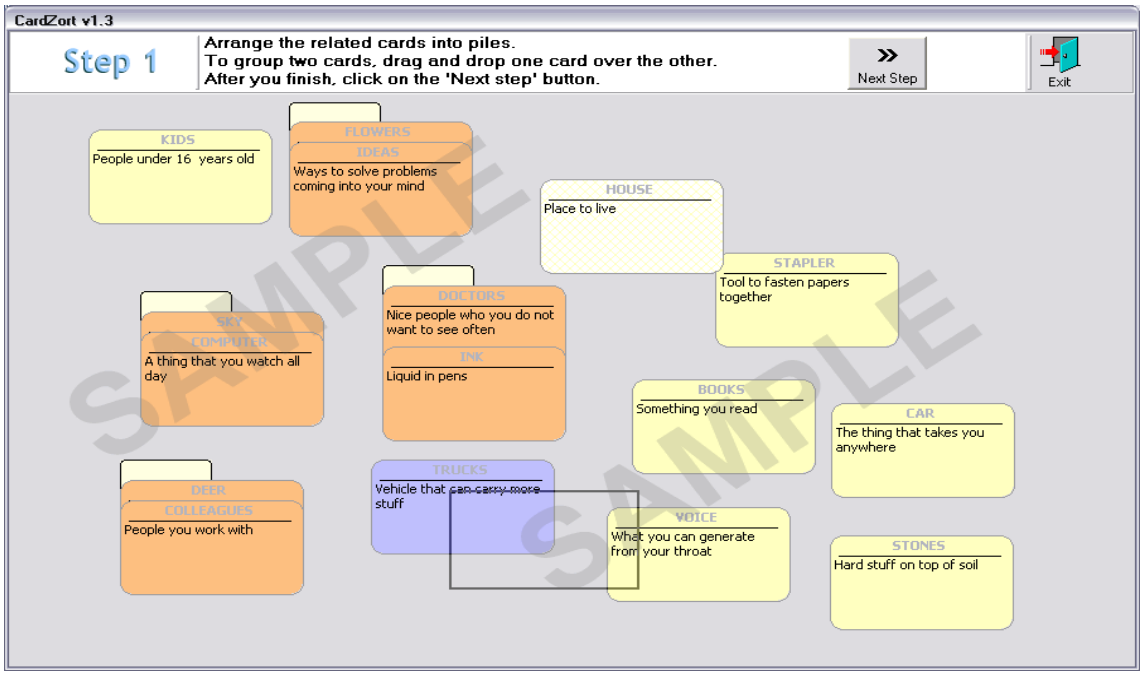


Figura 1.3.3-2 Creación de grupos de categorías.

El segundo paso es colocarle los nombres a cada uno de los grupos creados como se muestra en la siguiente figura:

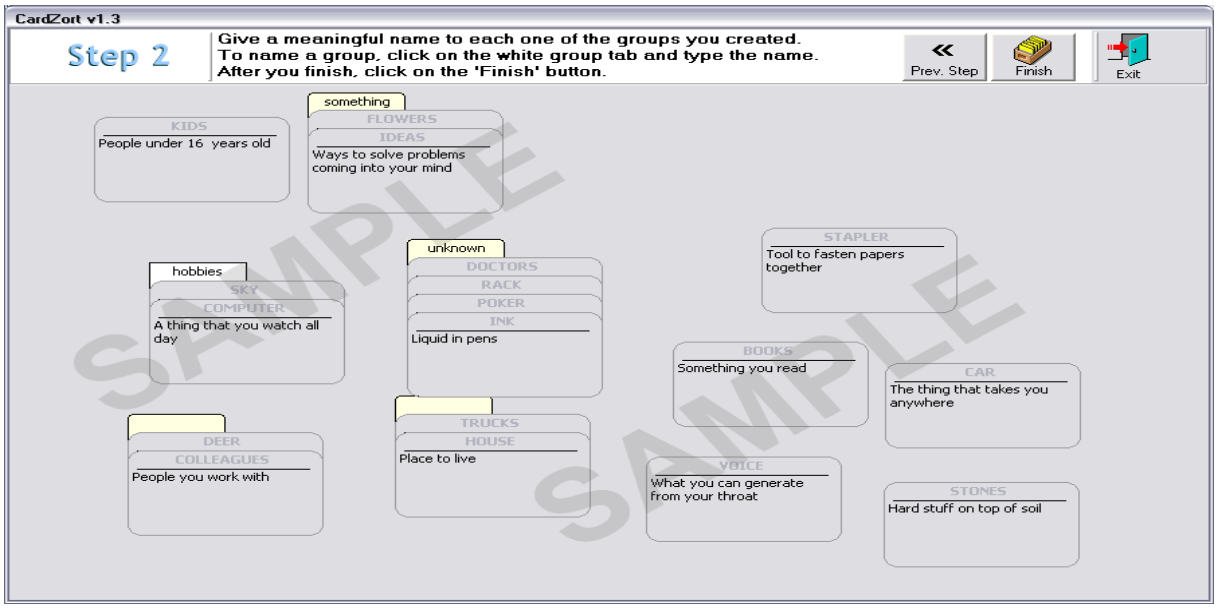


Figura 1.3.3-3 Títulos de las categorías.

Esta aplicación utiliza el CardCluster para realizar los análisis de los resultados obtenidos de la prueba, así como un análisis de las etiquetas revisando cuales son las preferencias de los usuarios sobre los posibles nombres de los grupos resultantes.

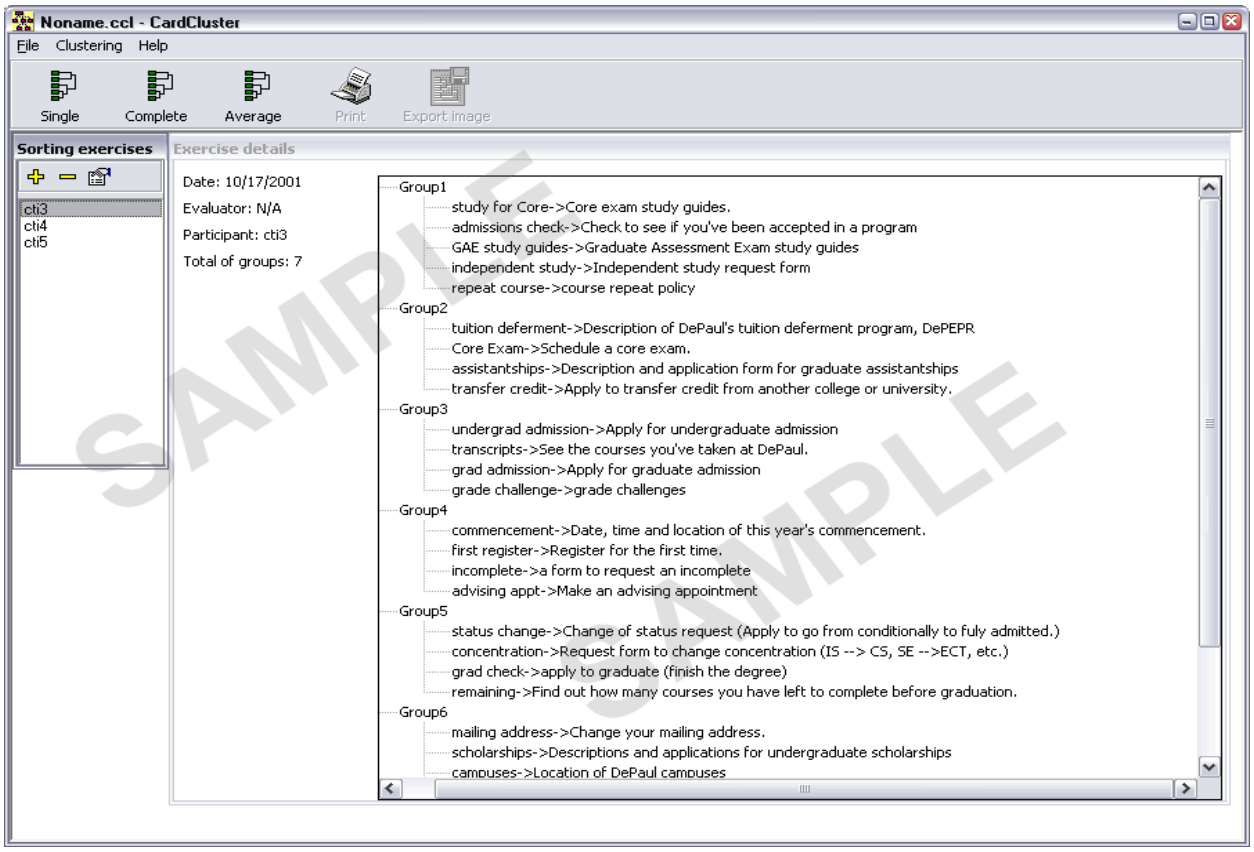


Figura 1.3.3-4 Portada principal del CardCluster.

Luego de culminado algún ejercicio CardZort genera un archivo de tipo texto con extensión ".Cz" Card Cluster toma estos archivos como insumo para realizar el análisis de los agrupamientos de las agrupaciones realizadas. Card Cluster solo hace posible la obtención de resultados por medio de un gráfico en forma de árbol como muestran la siguiente imagen:

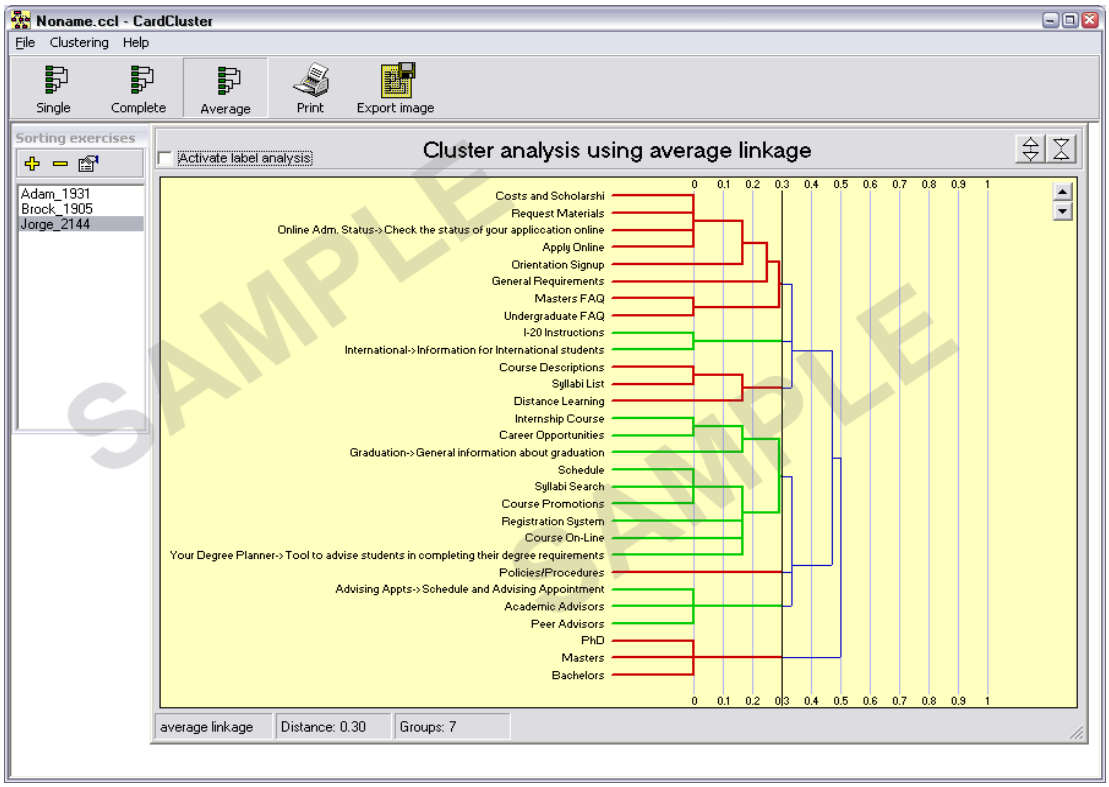


Figura 1.3.3-5 Análisis del Cluster utilizando "Average".

Por otra parte, el estudio de dicha aplicación nos ha llevado a conocer algunos detalles como la arquitectura sobre la cual se realizo dicho sistema, así como el conocimiento de flujo de datos del mismo. Las siguientes imágenes muestran la arquitectura y el flujo de datos del sistema en general.

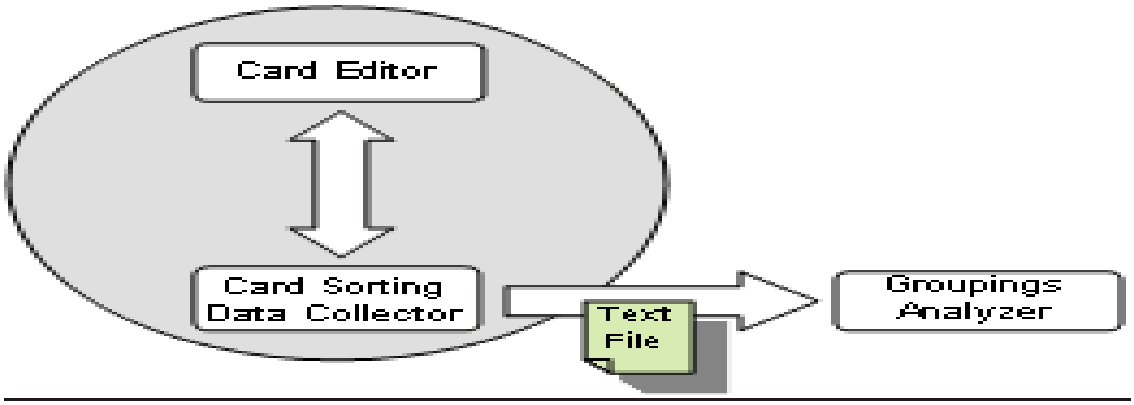


Figura 1.3.3-6 Arquitectura del sistema de CardZort.

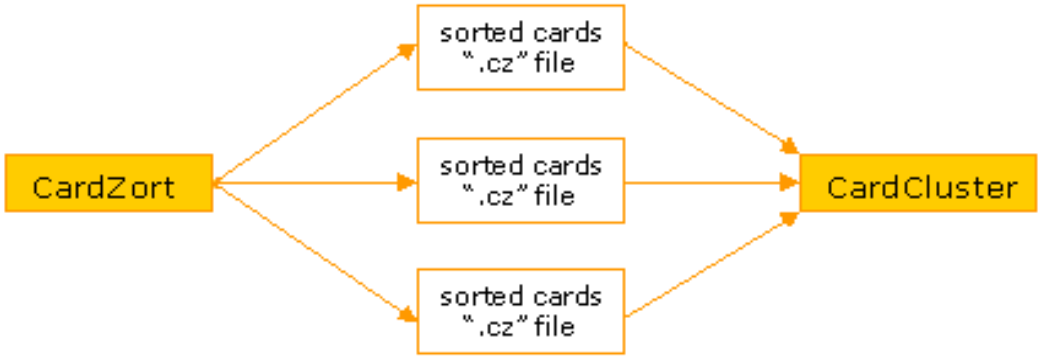


Figura 1.3.3-7 Flujo de datos entre CardZort y CardCluster

Finalmente se puede concluir que la aplicación tiene algunas ventajas, desde facilidades de trabajo, mejor obtención de resultados incluyendo también el ambiente gráfico en el que se realiza la prueba, el cual simula en muchos aspectos la verdadera selección de tarjetas. En el sitio http://www.cardzort.com/cardzort/index.htm se puede obtener un demo de este software que no realiza todas las funcionalidades empezando desde limitaciones existentes en el número de tarjetas a crearse entre otras. Pero es de mucha importancia resaltar que su principal desventaja consiste en *ser un software propietario*, por lo que su completo uso requiere del pago de una licencia. Otro punto negativo que se puede resaltar en dicho software es que no presenta dentro de sus funcionalidades ninguna otra de las técnicas matemáticas, como por ejemplo: el Análisis de Secuencia. Las características antes mencionadas, junto con el cambio que se esta realizando en la Universidad de las Ciencias Informáticas de comenzar a trabajar con herramientas creadas en la propia institución, hacen de la utilización de WebZort una opción poco factible.

1.4 Fundamentación de la de la metodología de desarrollo que se utilizará.

Antes de seleccionar una metodología de desarrollo de software, se hace necesario el conocer los costos de producción y el presupuesto con el que se cuenta para la realización del mismo, alcance del proyecto, tiempo estimado para evitar los retrasos y es importante conocer los recursos que se utilizarán para la creación del producto. Una vez conocidas estas especificaciones requeridas, entonces se procede a seleccionar el tipo de metodología que se utilizará. Toda metodología contiene el conjunto de procedimientos, técnicas, soporte documental que ayudaran al los desarrolladores a realizar un nuevo software, para que el mismo salga con mayor calidad, cumpliendo las expectativas de los usuarios, en el tiempo estimado y que pueda ser realizado de la forma más organizada posible. Con el objetivo de lograr mayor flexibilidad y que sea posible la realización de futuros cambios, se ha

elegido como metodología para el desarrollo del software, el *Proceso Unificado del Rational (RUP),* junto con la herramienta de Rational Rose y el UML como lenguaje de modelado visual del proyecto.

1.4.1 Proceso Unificado del Rational (RUP).

El Proceso Unificado es uno de los procesos de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. RUP como marco de trabajo genérico, puede ser aplicado a cualquier sistema de software, a cualquier producto que se este desarrollando y en cualquier área u organizaciones de producción.

El proceso unificado es un proceso de desarrollo de software basado en componentes que ayuda a mejorar la productividad del equipo de trabajo, definiendo actividades, roles y responsabilidades dentro del equipo de desarrollo, desde los jefes de proyectos a los analistas y desde los desarrolladores a los probadores. Define el cómo y el cuándo se realizará cada actividad definida hasta alcanzar la meta planteada. Utiliza el Lenguaje Unificado de Modelado (UML) para la construcción de todos los esquemas de un sistema de software. Además, se encarga del proceso verificar la calidad el software y controla los cambios que podrían realizarse. Ahora bien, existen tres características fundamentales que tiene el Proceso Unificado del Racional que lo resumen, estos son:

Dirigido por casos de uso: Los casos de uso se especifican y se diseñan, representan lo que los usuarios futuros necesitan y desean, estos enlazan los flujos de trabajo en el proceso de desarrollo del software. Los casos de uso son captados mediante el modelamiento del negocio y se representan a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo a partir de modelos que se obtienen como resultado de los diferentes flujos de trabajo, representando la realización de los casos de uso.

Centrado en la arquitectura: La arquitectura del software define la estructura, comportamiento y funcionalidad del mismo, describe los elementos del modelo que son mas importantes para su construcción como base para comprenderlo, desarrollarlo y producirlo económicamente, por lo que muestra la visión común del sistema completo en la que el equipo de trabajo y los usuarios del producto deben estar completamente de acuerdo. RUP se desarrolla mediante iteraciones, comenzando por los Casos de Uso relevantes desde el punto de vista de la arquitectura el cual se representa a través de vistas en las que se incluyen los diagramas de UML.

Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones dividiendo el trabajo en pequeñas partes o mini proyectos, donde cada uno de ellos es una iteración que concluye en un incremento. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en el flujo de

trabajo, y los incrementos al crecimiento del producto, por lo que las iteraciones constituyen una secuencia de actividades con un plan de actividades y un criterio de evaluación cuyos resultados son versiones diferentes del software.

1.4.2 Herramienta Rational Rose.

En la actualidad con el desarrollo de las tecnologías, las organizaciones se ven obligadas a trabajar mucho más rápido, crear productos de mayor calidad y en plazos de tiempos más cortos. Por esa razón, las herramientas de modelado se han vuelto cada día más importantes, a medida que los sistemas evolucionan y se vuelven más complejos, estas se tornan más significativas por los beneficios que proporcionan. El Racional Rose es una herramienta que soluciona estos problemas para el desarrollo de software, ya que cubre todo el ciclo de vida de un proyecto, desde la concepción y formalización del modelo, construcción de los componentes, transición a los usuarios hasta la certificación de las distintas fases y entregables. Usar una sola herramienta durante todo este ciclo, nos asegura que estamos construyendo el sistema de forma correcta. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción.

Rational Rose acelera la implementación al automatizar modelos arquitectónicos que se apoyan en las soluciones que ofrecen WinDNA, Interprise Java, Web y XML, además permite la adaptación y mejoras ante las condiciones cambiantes del negocio. Esta herramienta permite visualizar, entender y refinar toda la arquitectura y los requerimientos del producto antes de enfrentarse al código, lo cual constituye un ahorro de tiempo. Esta herramienta es considerada líder en el mundo para el modelamiento visual, ya que domina el análisis, diseño y construcción orientado a objetos. La International Data Corparation (IDC), le ha dado por 4 años consecutivos, desde 1998 al 2002 ese calificativo. Utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Permite que haya varias personas trabajando a la vez en el proceso de modelamiento, el cual está basado en el Unified Modeling Language (UML) como notación estándar para la arquitectura de software, lo cual hace que el Rational Rose permita a todo el grupo de desarrollo, la posibilidad de trabajar con una misma herramienta y bajo un mismo lenguaje. También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades. Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML. Proporciona mecanismos para realizar la denominada Ingeniería Inversa, a partir del código de un programa se puede obtener información de su diseño. En fin Rational provee a todos los diseñadores soporte UML incomparable, ingeniería para multi lenguajes, completo soporte al equipo y un desarrollo basado en componentes.

1.4.3 Lenguaje UML.

UML es un lenguaje unificado de modelado, de estandarización que permite documentar y desarrollar los elementos que forman parte de un sistema software orientado a objetos. Es la forma de representar diferentes perspectivas de un sistema y está compuesto por diversos elementos gráficos, los que combinados conforman los diagramas. Este lenguaje fue creado por Grady Booch, James Rumbaugh e Ivar Jacobson con el objetivo de crear una notación única para simplificar y consolidar el gran número de métodos de desarrollo orientado a objetos que habían surgido.

Hoy en día UML es una de las herramientas de gran importancia en el mundo del desarrollo de software y la naturaleza de su éxito radica principalmente en la naturalidad de su uso, pues la base de UML esta dada por los diferentes tipos de diagramas que dictan la manera en la que será diseñado un sistema. Es un concepto nuevo en el que la comprensión del problema en si resulta mucho más amigable tanto para los desarrolladores como para el cliente mismo, ya que conocerá de forma más detallada y explicada el producto que adquirirá.

1.5 Modelo de Arquitectura de tres Capas utilizado para la Implementación.

Durante mucho tiempo han venido desarrollándose estándares para integrar todo el ambiente de desarrollo, definir estructuras y métricas para garantizar una mejor calidad del software; esto ha sido llamado *Arquitectura del Software* la cual establece fundamentos y define una línea común para lograr alcanzar los objetivos del sistema de información y a su vez, cubrir todas las necesidades del software. Entre los tipos más comunes de arquitecturas se encuentra la *Arquitectura de tres capas*, que como expresa su nombre, está compuesta por tres capas, las cuales son: la capa de *interfaz o presentación*, la capa de *acceso a datos* y la capa de *reglas o lógica de negocio* la cual es realmente quien va a representar a la empresa obviando tanto la estructura de los datos como su ubicación. El objetivo principal de la utilización de esta arquitectura es el de lograr una separación de la lógica de negocios a la de la lógica del diseño. Constituye una arquitectura única para la creación de aplicaciones y para integrar todo el ambiente de desarrollo.

1. **Interfaz o Presentación**: Esta capa representa la parte del sistema con la que interactúa el usuario. Esta capa se comunica únicamente con la capa del negocio, también es conocida como *interfaz grafica* la cual debe ser entendible y fácil de usar para el usuario, ya que si este no puede interactuar con la aplicación de forma eficiente, no podrá realizar su trabajo y el éxito global de la misma podría estar comprometido.
2. **Lógica de negocio**: El comportamiento de la aplicación es definido por los componentes que modelan la lógica de negocio. Estos componentes reciben las acciones a realizar a través de la capa de presentación (a través de los usuarios), y

llevan a cabo las tareas necesarias utilizando la capa de acceso a datos para manipular la información del sistema. En esta capa es donde se establecen todas las reglas que deben de cumplirse. El hecho de tener la lógica de negocio separada del resto del sistema permite lograr una integración más sencilla y eficiente con otros sistemas externos.

3. **Acceso a datos:** Entre las funcionalidades que incluyen esta capa están: el almacenamiento, la actualización y la consulta de todos los datos contenidos en el sistema. Hablamos de acceso a datos cuando estamos manipulando lo que llamamos *tipo de datos persistentes.* Generalmente esta capa es un servidor de base de datos, aunque podría ser cualquier otra fuente de información. Esta capa puede estar en el mismo servidor que las de Presentación y Lógica del Negocio o en uno o incluso varios servidores.

1.5.1 Ventajas de la Arquitectura de tres capas:

Dicha Arquitectura permite la **reutilización** ya que la aplicación está compuesta por un conjunto de componentes que se comunican entre sí a través de interfaces y que cooperan para lograr el comportamiento deseado. Esto permite estos componentes puedan ser reemplazados por otros, y que también puedan ser utilizados en otras aplicaciones. Otra de las ventajas es el **crecimiento**, ya que cada uno de los componentes de la aplicación pueden ser separados, por lo que puede hacerse que proyectos de gran envergadura puedan dividirse en pequeños proyectos más simples pudiéndoseles agregar nuevos servicios según la medida de crecimiento de la organización. También tenemos el u**so eficiente del hardware** ya que como los componentes pueden ser divididos, no se necesitará de grandes servidores para el procesamiento de datos, sino que le problema se resolvería usando maquinas pequeñas, económicas y fáciles de reemplazar. Otra de las ventajas es que **encapsula los datos** ya que las aplicaciones cliente se comunican con los datos a través de peticiones que los servidores responden ocultando y encapsulando los detalles. Otras de las principales ventaja son las **distintas presentaciones** que se evidencia gracias a que separa la *presentación* de la *lógica de negocios*, por lo que es más sencillo realizar tantas presentaciones diferentes como dispositivos con capacidades e interfaces se tenga. Además, esta arquitectura **ahorra tiempo y costos** y **mejora la calidad en las aplicaciones** ya que como son construidas en unidades separadas, estas pueden ser probadas independientemente y más detalladas lo que posibilita obtener un producto mucho más sólido.

1.6 Lenguaje de Programación Java. Plataforma NetBeans 6.0.

Para el desarrollo del trabajo se escogió la tecnología Java porque es una tecnología madura, muy eficaz y sorprendentemente versátil, Permite a los desarrolladores crear software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma y en cualquier otro sistema operático. Java provee toda funcionalidad de un lenguaje potente, pero a la vez elimina las características menos usadas y más complejas de dichos lenguajes, de ahí y la seguridad que aporta. Otra de las razones importantes para escoger este lenguaje es la migración hacia Software libre en que se encuentra la Universidad en estos momentos y la necesidad de utilización de lenguajes no propietarios para el desarrollo de software, además del paso hacia el uso del sistema Operativo Linux hacia el cual se esta migrando en la UCI y en Cuba respectivamente.

La tecnología Java está compuesta por dos partes: El lenguaje de programación, y la plataforma de desarrollo.

1.6.1 Lenguaje de programación Java.

Algunos de los lenguajes de gran aceptación y en los que podríamos realizar la implementación del trabajo podemos encontrar el C++, PHP, y Java. Uno de los objetivos de este trabajo es realizar la implementación de una aplicación desktop, por lo que el uso de PHP no es el más recomendable ya que este se utiliza fundamentalmente para entornos Web.

C++ tiene una gran rapidez de ejecución ya que al compilarlo se genera el código objeto nativo de cada máquina, pero tiene una debilidad, y es que lo hace dependiente de un único sistema operativo, quitándole portabilidad al software final. Por otra parte C++ tiene librerías de clases propias del lenguaje y menos funcionales y otras que no son del lenguaje y que han sido creadas por terceros. Además, en C++ es el programador quien tiene que eliminar toda referencia de los punteros que ya no se estén utilizando, a diferencia de java, que cuenta con lo que se le llama "*Reciclador de memoria dinámica*", estas desventajas hacen que C++ no sea una oferta del lenguaje a tener en cuenta.

Una buena opción sería el lenguaje de programación Java, el cual es descrito como "simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico. (Peñalver, 2007).

Es ***simple***, ya que ofrece todas las funcionalidades de un lenguaje potente pero sin las características menos usadas y más confusas de éstos. Java fue creado con muchas de las características de C++ y como este es un lenguaje muy difundido hace que sea aceptado entre la comunidad de programadores e influya a su favor. Por otra parte tenemos la ventaja de que Java reduce más del 50% de los errores existentes en otros lenguajes como C y C++ entre los que se encuentran la aritmética de punteros, no existen referencias, necesidad de liberar memoria, entre otros.

Es ***Orientado a Objetos,*** ya que Java agrega algunas mejoras al lenguaje, además de que trabaja con sus datos como objetos y con interfaces a estos objetos, manteniendo las tres características fundamentales de la programación orientada a objetos: herencia, polimorfismo y encapsulamiento.

Java permite a los programadores acceder a la red con gran facilidad gracias a que se ha construido con extensas capacidades de interconexión TCP/IP y proporciona librerías y herramientas que permiten que los programas puedan ser **distribuidos**, siendo estas librerías medios de acceder e interactuar con protocolos como HTTP y FTP. Java surgió entones como el lenguaje de programación para entorno de red por excelencia. Esto explica porque Java fue pensado para ser seguro desde sus principios. Los programas escritos en Java no pueden ser atacados por virus, pues para que estos tengan efecto deben utilizar rutinas de acceso directo a memoria, que Java no tiene.

Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error, por lo que su **robustez** reduce drásticamente el tiempo de desarrollo de aplicaciones, ya que proporciona:

- Comprobación de punteros.
- Comprobación de límites de arreglos.
- Excepciones.
- Verificación de código objeto o código de máquina.

Java posee una **arquitectura neutral**, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-*time)* puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado. Por otro lado, las aplicaciones de Java resultan extremadamente **seguras**, ya que no acceden a zonas delicadas de memoria o de sistema, con lo cual evitan la interacción de algunos virus. El código de Java pasa por varias pruebas antes de ser ejecutado en alguna máquina y si no se genera ningún mensaje de error, entonces sabemos que no hay intentos de violaciones de código ni problema con el manejo de los datos entre otros. Además, para evitar modificaciones por parte de los crackers de la red, implementa un método ultra seguro de autentificación por clave pública. El Cargador de Clases puede verificar una firma digital antes de realizar una instancia de un objeto. Por tanto, ningún objeto se crea y almacena en memoria, sin que se validen los privilegios de acceso.

Otra de las ventajas que encontramos en el lenguaje es la capacidad de realizar ***varias tareas a la vez***; esto significa que un programa puede dividirse en varios fragmentos para ser más eficiente en realiza su tarea.

1.6.2 La plataforma NetBeans 6.0.

La plataforma Java tiene dos componentes: La máquina virtual de Java (JVM) y el API Java (Application Program Interface). Java es compilado a un código intermedio o bytecode, el cual es interpretado por una máquina virtual de Java. Esta hace posible que una aplicación que haya sido implementada en Java se ejecute en cualquier sistema operativo con soporte para la máquina virtual. Esta máquina virtual proporciona un entorno de ejecución que convierte el código neutro de Java al código nativo del ambiente en que este siendo ejecutada. La máquina virtual de Java es la base de la plataforma Java, es llevada a diferentes plataformas de hardware.
Para los programadores el API de Java le proporciona muchas utilidades por la gran colección de componentes de software que le brinda. Incorpora muchos aspectos como ejecución remota, componentes, seguridad, acceso a bases de datos, etc.

Para realizar la aplicación, se ha decidido utilizar el **IDE** (Integrated *Development Environment* o Entorno de Desarrollo Integrado) NetBeans 6.0, ya que es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Esta escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso y el desarrollador obtiene todas las herramientas que necesita para crear aplicaciones profesionales de escritorio, empresariales y móviles. Este IDE corre sobre varias plataformas y es muy fácil de instalar. Creado por Sun Microsystems, para los sistemas operativos *Windows, GNU/Linux, Mac OS X y Solaris.*
NetBeans es un entorno de desarrollo integrado, modular, basado en estándares, se refiere a una plataforma de aplicaciones que puede ser usada como un framework genérico para construir cualquier tipo de aplicaciones. Permiten que estas aplicaciones puedan ser desarrolladas a partir de un conjunto de componentes de software llamados módulos los cuales contienen clases de Java escritas para interactuar con las APIs de NetBeans. La versión 6.0 incluye mejoras significativas y nuevas características, incluyendo una infraestructura completamente re- escrita y soporte para idiomas adicionales, incluyendo una aunque en la actualidad aun no cuenta con el español.
Esta plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole a los desarrolladores enfocarse en la lógica específica de su aplicación. Entre las mejoras que presenta esta plataforma se encuentran:

- La administración de la interfaz de usuario.
- Administración de las configuraciones del usuario.
- Administración del almacenamiento (guardando y cargando los datos).
- Administración de ventanas.

Por otra parte NetBeans 6.0 posee mejoras en el editor, entra las que se encuentran el completamiento de código, que ahora es mucho más rápido. El editor posee una lista posible de palabras claves,

atributos o campos vulnerables, así como también se puede realizar un mejor resaltado de código. En cuanto a su instalación… esta es mucho más fácil, pudiendo elegir e instalar las características necesarias y requeridas, logrando obtener las funcionalidades que se necesitan, todo en un solo lugar. Posee asistentes para la escritura de servicios Web, diseñadores para interfaces gráficas para Swing y JSF, así como soporte para librerías de terceros MyFaces, Tomahaw, entre otras.

1.7 Conclusiones.

Como se ha estado viendo en este capítulo, se han analizado una gran variedad de software y sistemas, que automatizan la técnica de la Arquitectura de Información *Card Sorting*, viendo soluciones muy buenas, pero que no se adaptan a las necesidades ni a la política de independización tecnológica y software libre que se está aplicando en la Universidad en estos momentos. Por otra parte, vemos que de las aplicaciones estudiadas, ninguna utiliza el A*nálisis de Secuencia* en sus soluciones, por lo que eso constituye también una desventaja en general. En el capítulo se han estudiado las diferentes tecnologías, herramientas y tipo de arquitectura que se utilizarán para darle cumplimiento a los objetivos trazados en comienzo del trabajo.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

Para comenzar con el proceso de desarrollo del software, es necesario conocer si es factible seguir adelante con el proyecto, por lo que hay que alcanzar un buen conocimiento del problema en cuestión. En este capítulo veremos el problema y la situación problémica de forma detallada, se realizará una descripción sobre el objeto de estudio del trabajo y se conformara el modelo de negocio. También se realizará la propuesta del sistema, explicando previamente los procesos del negocio. Se definirán los requisitos funcionales y no funcionales, y de los primeros se obtendrán los casos de usos del sistema que serán descritos para lograr una mejor comprensión de su funcionamiento, de la misma manera se definirán los actores que interactúan con los mismos.

2.2 Objeto de Automatización.

La idea que se persigue con este trabajo es facilitar el trabajo de los Arquitectos de Información en la Universidad de las Ciencias Informáticas. Se parte del estudio realizado sobre los diferentes sistemas que automatizan la técnica matemática de la Arquitectura de Información *Card Sorting*, pero no es un objetivo del trabajo automatizar dicha técnica tal y como lo hacen los sistemas estudiados en el primer capítulo, sino automatizar la técnica antes mencionada e incluir las funcionalidades que brinda otro de los modelos matemáticos: el *Análisis de Secuencia*, obteniéndose una aplicación que; aunque más sencilla, brindará una vía más segura y rápida para la realización del trabajo de nuestros diseñadores, facilitará todo el proceso de búsqueda de información para obtener mejores resultados en la creación de sitios Web más navegables, *eficientes* y *accesibles*, con diseños más centrados a lo que desean los usuarios finales del producto.

2.3 Propuesta del sistema.

Para la creación de dicho sistema, se pretende automatizar las técnicas matemáticas de co-ocurrencia Card Sorting y Análisis de Secuencia. Se realizará una aplicación servidor y otra cliente conectadas mediante el protocolo TCP/IP. En la aplicación servidora se crearán los estudios para luego ser publicados y acceder a los mismos mediante la aplicación cliente. Cada estudio creado contendrá la lista de tarjetas que se irán creando y permitirá seleccionar cuál de estas dos técnicas es la que se realizará (Card Sorting o Análisis de Secuencia), también tendrá una descripción sobre qué se tratará en el mismo. Se adicionarán uno por uno los diferentes artículos que formarán parte del grupo de

tarjetas a las que se les realizará la selección y en caso de que sea necesario, cada uno de estos artículos llevará una descripción corta en la parte posterior para que se entienda de qué se trata dicho artículo. Cada tarjeta representa cada contenido y funcionalidad que tendrá el producto Web y deberá tener un número (id), para facilitar el análisis una vez realizado el sorteo. El sistema debe ser capaz de guardar un archivo que tendrá la lista de todos los estudios creados la cual debe poder visualizarse y debe modificarse, para en caso de necesidad, incluir o eliminar alguno de estos o modificar los artículos o tarjetas pertenecientes a cada estudio.

Una vez creado el estudio, la aplicación cliente puede ejecutarse para que los usuarios se conecten al servidor para realizar la prueba. El sistema deberá permitir la creación de grupos de categorías para ubicar las selecciones de tarjetas que realizarán por los usuarios (esto lo permitirá para todos los realizadores de la prueba), En la aplicación servidora en el momento que se crea el estudio si se desea realizar la técnica de *Card Sorting* de tipo "cerrado" entonces el sistema deberá permitir crear los grupos que serán los predeterminados. En caso de ser del tipo "abierto", el cliente en la aplicación destinada al mismo, será capaz de crear los grupos describirlos y nombrarlos. El sistema deberá permitir ir salvando el trabajo de manera que los usuarios que van realizando la prueba puedan ir subiendo los cambios que vayan haciendo en su selección hacia la aplicación servidora, la cual irá almacenando y guardando los cambios que vaya realizando cada usuario durante todo el tiempo que esté corriendo; una vez que se detenga la aplicación servidora ya los usuarios no podrán realizar más cambios en su estudio. El sistema deberá cargar todas las agrupaciones que hayan realizado y subido a la aplicación cliente los usuarios y permitir la ejecución del algoritmo de Card Sorting, para luego devolver cuales serán los grupos de contenidos que conformarán el sitio Web deseado, así como el nombre o etiquetado que llevarán los mismos.

Una vez obtenidos estos resultados, y luego de haber seleccionado realizar el Análisis de Secuencia, es necesario que el sistema compruebe que se haya realizado con anterioridad la selección de tarjetas (Card Sorting), por lo que debe permitir poner a correr la aplicación servidora y la aplicación cliente deberá cargar el nombre de los grupos de categorías devueltos por el sorteo de tarjetas y gestionará la manera en que cada usuario realizará el ordenamiento de estos grupos así como el ordenamiento de las tarjetas pertenecientes a cada uno de los grupos, según estime conveniente. De igual manera deberá permitir subir los ordenamientos de cada uno de los usuarios participantes en el estudio. El sistema deberá permitir la ejecución del algoritmo de Análisis de Secuencia y después de esto, deberá devolver cual será el orden definitivo de los grupos de categorías que tendrá determinado sistema de información así como el orden de la información perteneciente a cada uno de los grupos.

2.4 Modelo del Negocio.

La modelación del negocio se realiza para lograr un entendimiento común de la organización entre los desarrolladores y los interesados en el proyecto. Lo primero a realizarse es la definición de los procesos que se automatizarán. Estos procesos estarán sujetos a un conjunto de reglas que también tendrán que ser definidas durante todo el modelado del negocio, por lo que se pretende con el modelado del negocio, es describir cada uno de los procesos del negocio, especificando distintos datos, como son las actividades, roles y las reglas del negocio. Y como objetivo general de este modelado es comprender la estructura y dinámica del medio en el cual se utilizará el sistema, asegurar que los usuarios finales del producto tengan un entendimiento común de lo que se está desarrollando y lo más importante, poder determinar los requerimientos del sistema necesarios. De esta manera se logra que antes de realizar el proceso de análisis y diseño de una herramienta informática se tengan definidos los procesos que funcionan correctamente y que deben ser automatizados.

2.4.1 Actores y Trabajadores del negocio.

Un Actor del negocio es cualquier individuo, grupo, entidad, organización o sistema de información externo que interactúa con el negocio para beneficiarse de los resultados.

Actor del negocio	Justificación
Usuario final	Es la persona que realizará el sorteo de un grupo de tarjetas y posteriormente entregara estos resultados al arquitecto de información.

Tabla 1 Justificación de los actores del negocio.

Un trabajador del negocio es una abstracción de una persona (o un grupo de personas), una maquina o un sistema automatizado; que actúa en el negocio realizando una o varias actividades. Representa un rol y manipula entidades del negocio. Interactúa con otros trabajadores del negocio.

Trabajador del negocio	Justificación
Arquitecto	Es la persona encargada de crear las tarjetas que se utilizarán para la selección así como será el

	responsable de la obtención de los resultados del estudio realizado.

Tabla 2 Justificación de los trabajadores del negocio.

2.4.2 Diagrama de Casos de uso del negocio.

DIAGRAMA DE CASOS DE USO DEL NEGOCIO.

2.4.3 Descripción de los casos de uso del Negocio.

Nombre del CU del negocio:	**Aplicar Estudio**
Actor del negocio:	Usuario Final.
Trabajador del negocio:	Arquitecto.
Resumen: Se inicia cuando el Arquitecto de información crea las tarjetas con todo el contenido y sita a usuarios potenciales del producto para aplicarles el estudio y termina con la culminación del mismo.	
Acción del actor	**Respuesta del negocio**
	1- El arquitecto escoge el tipo de estudio que va a aplicar, crea las tarjetas y las entrega a los usuarios.

2- El usuario analiza las tarjetas y el contenido de las mismas.	
3- Conforma los grupos por afinidad o realiza el ordenamiento, luego entrega el estudio al arquitecto.	
Mejoras:	Al automatizar este proceso se podrán imprimir todas las tarjetas necesarias para la realización del estudio, así como facilitar la creación de grupos de categorías.

Tabla 3 Descripción del caso de uso del negocio: Aplicar estudio.

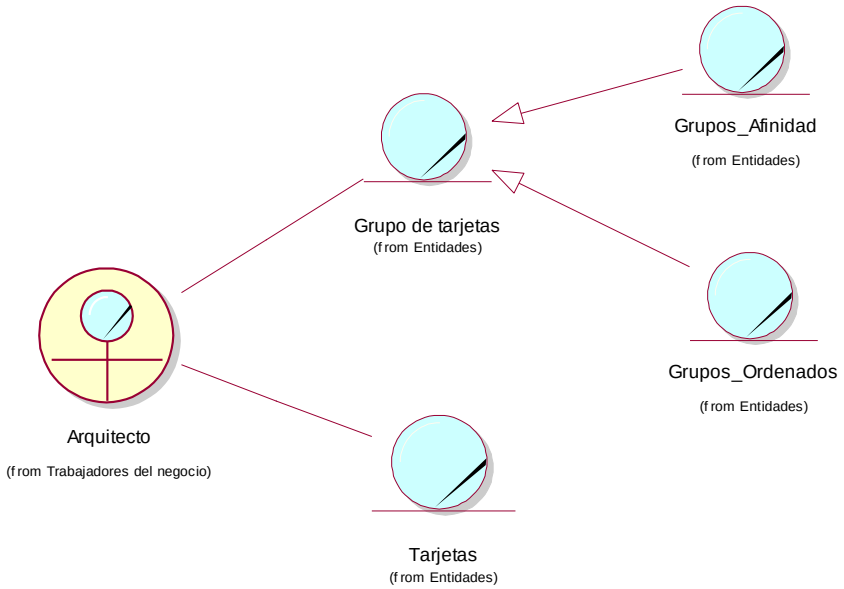
Nombre del CU del negocio:	**Analizar Resultados**
Actor del negocio:	Usuario Final.
Trabajador del negocio:	Arquitecto.
Resumen: Se inicia cuando el usuario final del producto culmina el estudio y entrega los resultados al arquitecto el cual se encargará de realizar el análisis de estos resultados, culminando dicho caso de uso con la obtención de resultados más óptimos.	
Acción del actor	**Respuesta del negocio**
1- El usuario culmina es estudio y lo entrega al arquitecto	2- El arquitecto recoge los datos del estudio.
	3- Realiza el algoritmo correspondiente al tipo de prueba (Card Sorting o Análisis de Secuencia).
	4- Obtiene los resultados finales del estudio.
Mejoras:	Con la automatización de este proceso, se espera facilitar la obtención de los resultados finales del estudio. Que estos resultados sean más rápidos y confiables.

Tabla 4 Descripción del caso de uso del negocio: Analizar Resultados.

2.4.4 Modelo de Objeto.

El modelo de objetos representa la relación existente entre los trabajadores del negocio y las entidades del mismo. Es el conjunto básico de objetos involucrados en el negocio, usualmente son modelados los objetos de la realidad.

MODELO DE OBJETOS.



2.4.5 Diagramas de actividades.

Un diagrama de actividades es un proceso que escribe cuales serán el orden de las tareas y actividades que se realizarán para lograr los objetivos del negocio. (Definidos en los casos de uso del negocio). Representa un paso en el flujo de trabajo o ejecución de una operación o grupo de actividades que se realizan para completar dicha operación.

DIAGRAMA DE ACTIVIDADES DEL CU: APLICAR ESTUDIO.

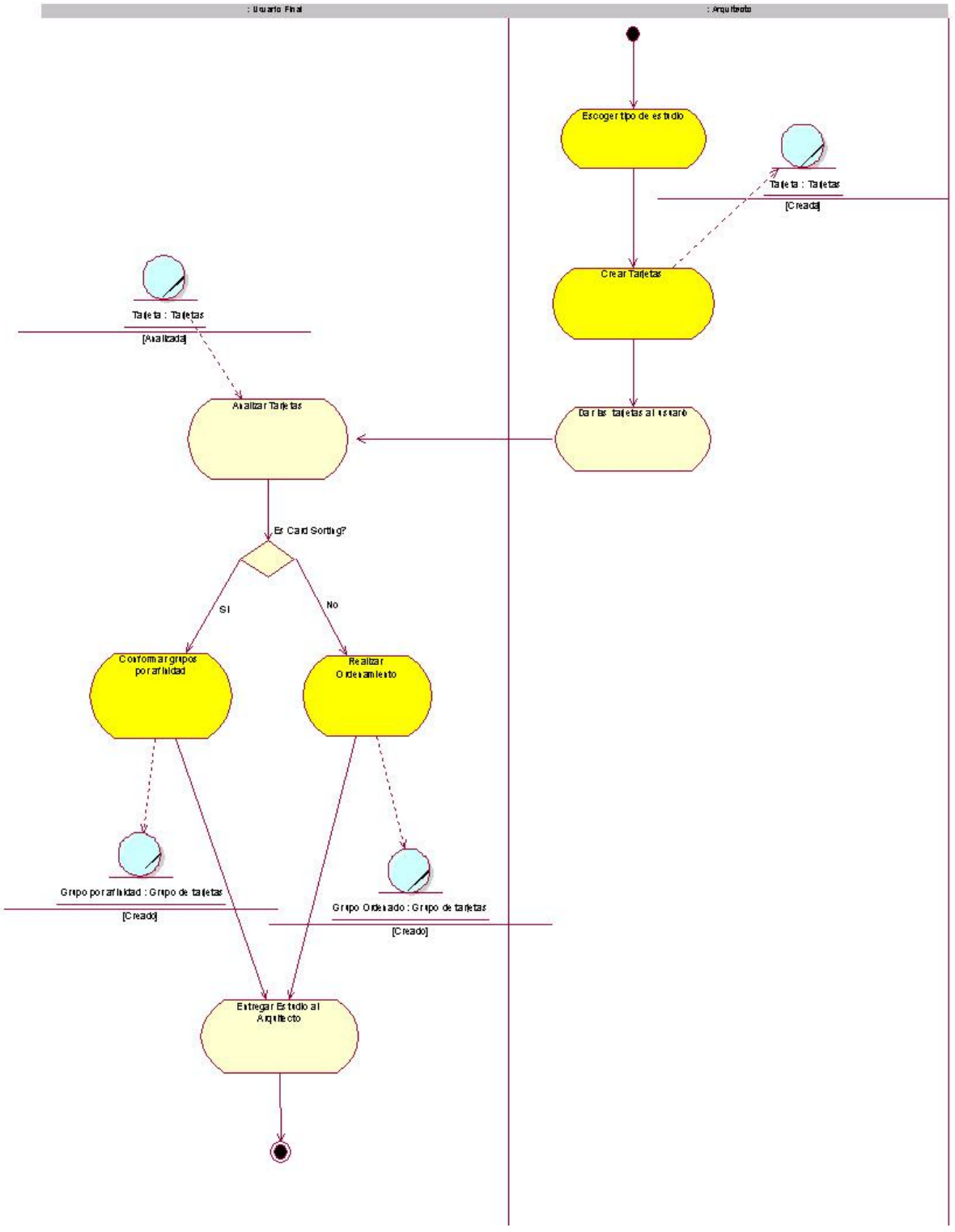
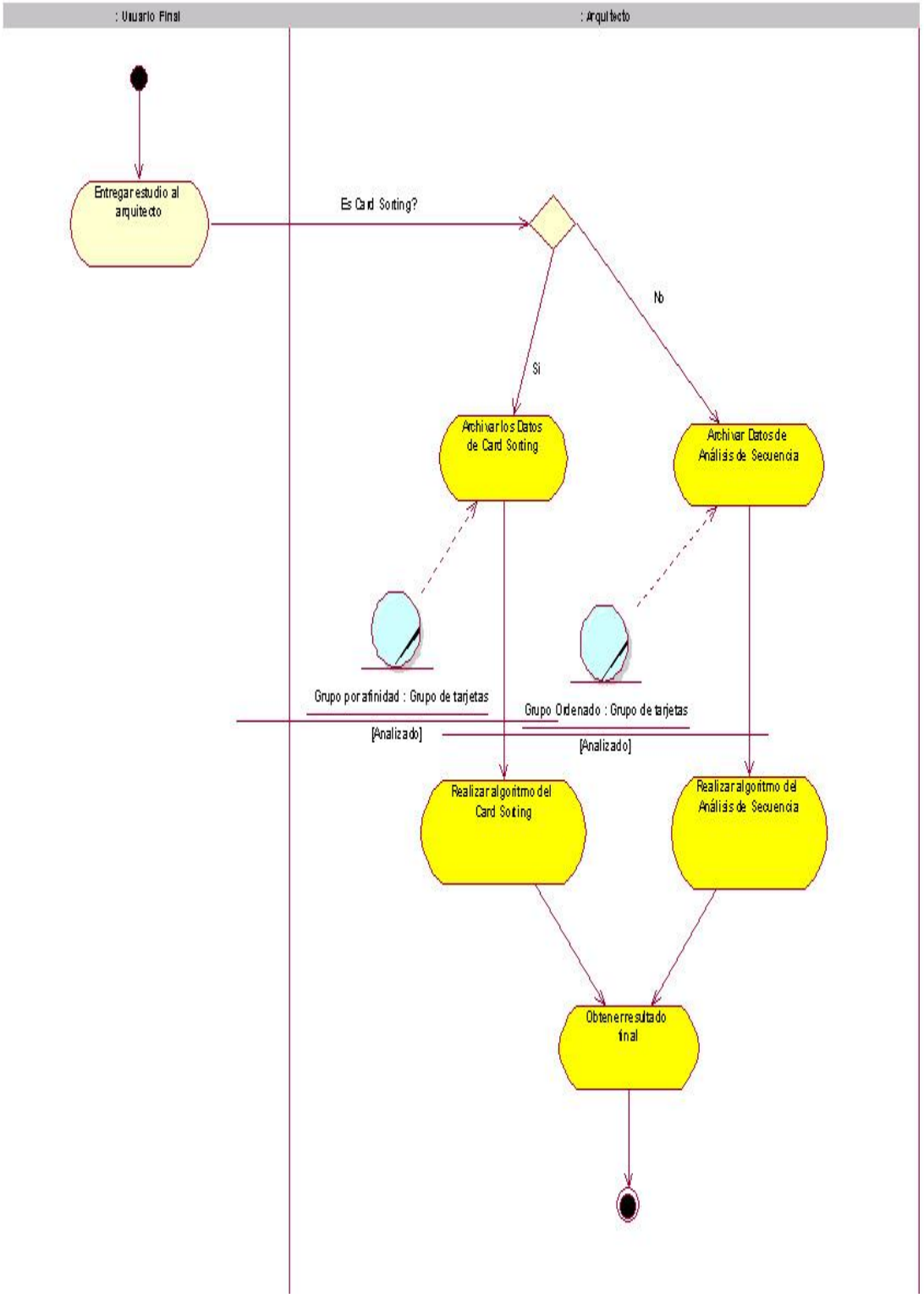


DIAGRAMA DE ACTIVIDADES DEL CU: ANALIZAR RESULTADOS.



2.5 Especificación de los requisitos de software.

Los requerimientos son muy importantes en el proceso de desarrollo de software, ya que determinan en gran medida el éxito del mismo. El objetivo de estos requerimientos es generar las especificaciones correctas que describan con la mayor claridad posible el comportamiento del sistema. A continuación se detallarán los requisitos *funcionales* y los *No funcionales* del sistema en desarrollo.

2.5.1 Requerimientos Funcionales.

Estos constituyen las capacidades o condiciones que el sistema debe cumplir.

- RF 1. Gestionar tarjetas.

  a) Conformar las tarjetas.

  - Nombrar las tarjetas.
  - Realizar descripción.

  b) Modificar tarjetas (modificar nombre y descripción).

  c) Mostar listado de tarjetas.

  d) Eliminar tarjetas de la lista.

- RF 2. Gestionar Estudio.

  a) Conformar estudio
  - Nombrar estudio.
  - Realizar descripción.

  b) Modificar estudio.
  c) Eliminar estudio.
  d) Mostrar el listado de los estudios creados.

- RF 3 Publicar Estudio.

  a) Permitir ver las direcciones IP y los nombres de los usuarios que realizan el estudio.

  b) Cambiar contraseña.

  c) Especificar puerto.

  d) Comenzar conexión.

- RF 4 Establecer Conexión.

  a) Permitir especificar IP de la PC servidora.

  b) Permitir introducir la contraseña del estudio para acceder al mismo.

  c) Permitir introducir el nombre del usuario que realizará la prueba.

  d) Permitir conectarse y detener la conexión, así como enviar el estudio terminado a la aplicación servidora y salir de la aplicación.

- RF 5. Realizar agrupaciones.

  a) Crear Grupos.

     - Nombrar grupos.

  b) Eliminar Grupos.

  c) Modificar Grupos

  d) Mostrar listados de grupos y listado de artículos por grupos.

- RF 6. Realizar ordenamiento.

  a) Permitir ordenar un grupo de tarjetas.

  b) Modificar ordenamiento.

  c) Verificar la existencia de grupos.

- RF 7. Realizar análisis de co-ocurrencias.

  a) Ejecutar el algoritmo de Card Sorting.

  b) Ejecutar el algoritmo de Análisis de Secuencia.

  c) Visualizar los resultados referentes a los dos

- RF 8. Mostrar resultados.

  a) Visualizar los resultados referentes al estudio de Card Sorting.

  b) Visualizar los resultados referentes al estudio de Análisis de Secuencia.

2.5.2 Requerimientos no Funcionales.

Las propiedades o cualidades que el producto final debe tener, son llamadas requerimientos no funcionales del software. Especifican propiedades del sistema como restricciones de ambiente y desarrollo, rendimiento, dependencias de plataformas, mantenimiento y confiabilidad. Estas cualidades hacen que el producto sea más atractivo, rápido y confiable.

- **Requerimientos de apariencia o interfaz externa.**

  - La herramienta propuesta será usada por personas que tienen conocimientos profundos de informática (arquitectos de información) por lo que necesita una interfaz amigable y colores estándares como el verde, blanco, gris y azul principalmente ya que son colores sencillos, no afectan demasiado la vista.

- **Requerimientos de usabilidad.**

  - La aplicación tendrá una buena aceptación ya que los usuarios que la utilizarán estarán muy vinculados con la interfaz y todo el contenido que gestiona la misma. Los usuarios podrán utilizar solo las funcionalidades que brinde la aplicación cliente, y los arquitectos podrán mediante la aplicación servidora tener todas las funcionalidades sin restricciones, ya que estas persona estarán especializadas en temas informáticos en especial personas relacionadas con la Arquitectura de información y específicamente arquitectos de información.

- **Requerimientos de rendimiento.**

  - El producto será más eficiente en la medida que se haga un mejor aprovechamiento de los recursos que se disponen. La aplicación debe ser rápida en los procesos de exportación de los ficheros por parte de la aplicación cliente hacia la servidora así como a la hora de efectuar los algoritmos de Card Sorting y Análisis de Secuencia.

- **Requerimientos de soporte.**

  - Para garantizar el soporte de esta herramienta, en caso de surgir algún problema el usuario tendrá la posibilidad de consultar la ayuda de la herramienta, además de que tendrá la posibilidad de localizar al desarrollador de la herramienta ya sea por correo o teléfono y enviarle las dudas en caso de que se presente cualquier problema con la aplicación y en caso necesario proporcionarle mantenimiento.

- **Requerimientos de portabilidad.**

  - El sistema podrá ejecutarse sobre cualquier sistema operativo. Para su desarrollo se usará la herramienta NetBeans IDE 6.0.

- **Requerimientos de funcionalidad.**

  - La aplicación tendrá el número de ventanas necesarias para que el sistema funcione correctamente, evitando las confusiones y ventanas abiertas de más.

- **Requerimientos de software.**

  - Para hacer uso de la aplicación se necesitará de una computadora personal cliente con el sistema operativo Windows 2000 o superior y una computadora servidora, con iguales características que la PC cliente. En ambas debes estar la maquina virtual de Java.

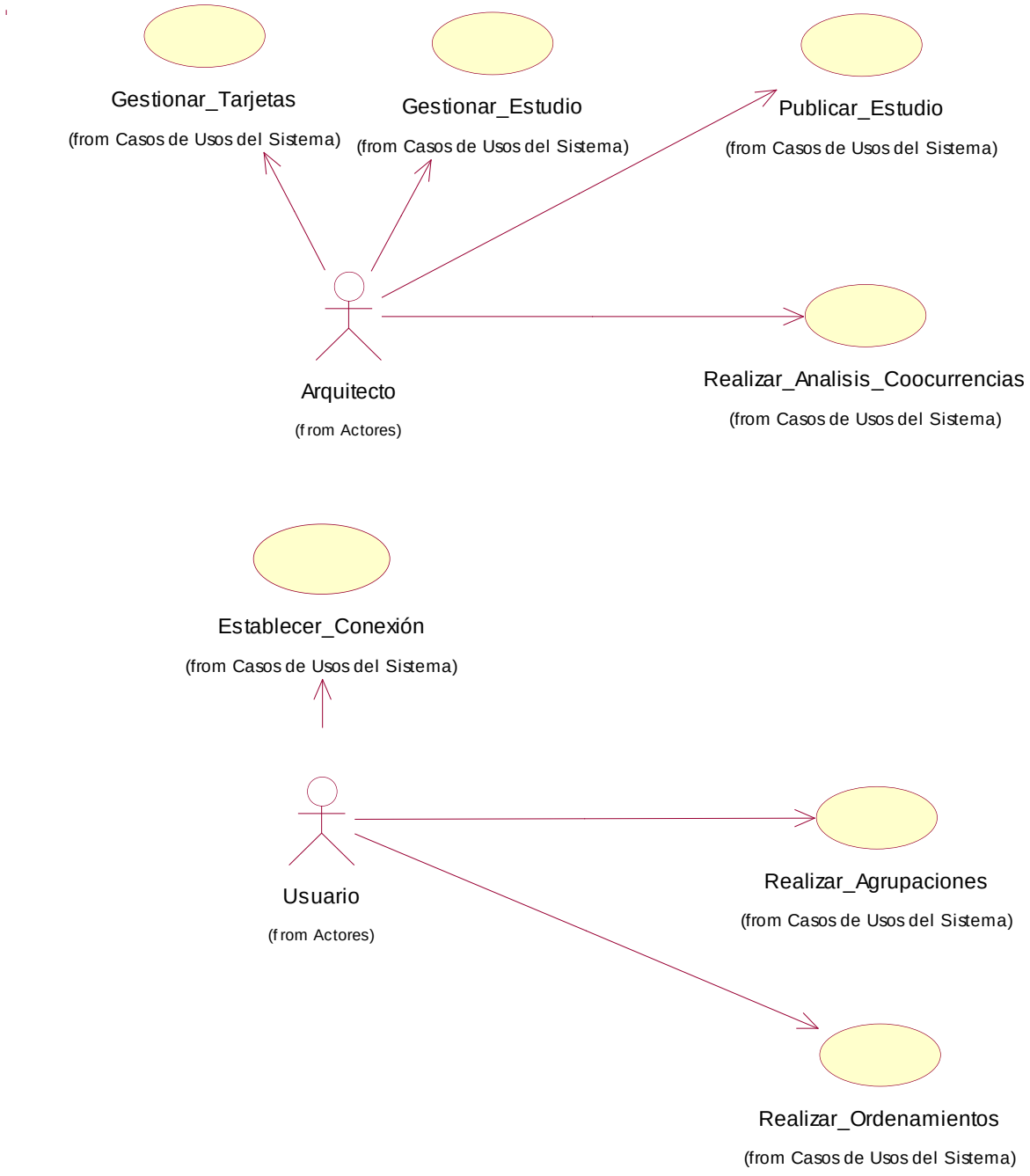
- **Requerimientos de Hardware.**

La herramienta deberá ser utilizada por medio de una computadora servidora y una PC cliente con 256 Mb de RAM como mínimo cada una. Y de capacidad en el disco duro, 40 Gb o más.

2.6 Diagrama de casos de uso del sistema.

Este diagrama muestra la funcionalidad y el comportamiento del sistema mediante su interacción con los usuarios. Representa los actores del sistema, los casos de uso y las relaciones existentes entre ellos.

DIAGRAMA DE CASOS DE USO DEL SISTEMA



2.7 Modelo de los casos de uso del Sistema.

2.7.1 Definición de los actores del sistema.

Actor del sistema	Justificación
Arquitecto.	Es la persona encargada de gestionar todo el proceso de creación de tarjetas, conformación de grupos y ordenamiento de tarjetas. Crea, elimina y modifica tarjetas, grupos de tarjetas y agrupaciones. Lleva a cabo la ejecución de los algoritmos Card Sorting y Análisis de Secuencia. Es la persona de realizar las salvas y guardado de los datos, así como su futuro uso.
Usuario.	Es el encargado de realizar las agrupaciones de contenidos y los ordenamientos de las tarjetas.

Tabla 5 Justificación de los usuarios del sistema.

2.7.2 Descripción de los casos de uso del sistema.

Por medio de los casos de uso del sistema se captura información de cómo un sistema o negocio trabaja, o cómo se quiere que trabaje. Describen la funcionalidad del sistema independientemente de la implementación.

A continuación se muestran las tablas con la descripción extendida de los casos de uso.

2.7.2.1 Caso de uso: Gestionar el proceso de creación de tarjetas.

Caso de uso	
CU-GT	CU_GestionarTarjetas
Propósito	El propósito de este caso de uso es el de crear las tarjetas que se usarán en el estudio.
Actores	Arquitecto
Resumen	El caso de uso se inicia cuando el arquitecto escoge la opción de crear una nuevo grupo de tarjeta y el sistema mostrará una ventana para poner nombre y descripción de las mimas, e irá mostrando la lista de las tarjetas que se vayan creando.
Referencias	RF 1
Precondiciones	
Pos condiciones	Se crea una lista de tarjetas que serán utilizadas en el estudio.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El Arquitecto selecciona en el menú de la interfaz principal la opción de crear grupo de tarjetas. (ver anexo no. 1)	2. El sistema muestra una ventana para la entrada de datos, como: 1- Un campo para mostrar la lista con los nombres de las tarjetas.

	2- Un campo para adicionar el nombre de las tarjetas 3- Un campo para realizar las descripciones de las tarjetas. 4- Muestra las opciones (botones) de adicionar, eliminar, aplicar y uno para mostrar la ayuda. - Permite modificar los datos de las tarjetas. 5- Para salvar dicho grupo de tarjetas, hay un campo dedicado al nombre del grupo y otro para realizar una descripción del mismo.
3. El Arquitecto selecciona la opción que desees (adicionar, eliminar una tarjeta, ver ayuda o modificar un grupo de tarjetas) a) Si selecciona adicionar. Ver "Sección Adicionar. b) Si selecciona eliminar. Ver "Sección Eliminar". c) Si se va a modificar. Ver “Sección Modificar”. d) Si se quiere ver la ayuda. Ver "Sección Ayuda".	
4. El Arquitecto escribe el nombre del grupo de tarjetas en el campo destinado al nombre.	
5. El Arquitecto en caso de creerlo necesario realiza una descripción del grupo.	
6. El Arquitecto presiona el botón de aceptar.	7. El sistema verifica que no exista otro fichero con ese nombre
	8. El sistema crea el fichero en la dirección especificada y muestra nuevamente la interfaz principal.
	9. Finaliza el caso de uso.
Sección: Adicionar	
Acción del actor	**Respuesta del sistema**
	10. El sistema crea una tarjeta con un nombre por defecto en el campo donde se muestra los nombres de las tarjetas.
11. El usuario selecciona (marca) el nombre de la tarjeta por defecto	
12. El usuario escribe el	

nombre de la tarjeta que quiere crear en el campo "nombre de la tarjeta"	
13. El usuario realiza la descripción de la tarjeta en el campo "descripción"	
14. El usuario selecciona el botón de "aplicar"	15. El sistema valida que no haya ninguna otra tarjeta con ese nombre creado
	16. El sistema cambia el nombre por defecto por el nombre que fue adicionado en el campo donde se muestran la lista de tarjetas.
Sección: Eliminar	
Acción del actor	**Respuesta del sistema**
17. El arquitecto selecciona una de las tarjetas de la lista	
18. El Arquitecto presiona el botón para eliminar tarjetas (botón eliminar)	19. El sistema muestra el mensaje "¿Realmente desea eliminar la tarjeta seleccionada?"
20. El Arquitecto selecciona el botón de aceptar	21. El sistema elimina la tarjeta antes seleccionada de la lista de tarjetas.
	22. El sistema muestra el listado de tarjetas modificado (sin la que se eliminó).
Sección: Modificar	
Acción del actor	**Respuesta del sistema**
23. El Arquitecto selecciona en el menú de la interfaz principal la opción de modificar un grupo de tarjetas.	24. El sistema muestra la interfaz para gestionar las agrupaciones de tarjetas.
25. El Arquitecto selecciona una de las tarjetas que se encuentran en la lista de tarjetas	26. El sistema muestra el nombre y la descripción de la tarjeta seleccionada.
27. El arquitecto en el campo del nombre, modifica el nombre de la tarjeta y en el campo de la descripción modifica la descripción de la misma.	
28. El arquitecto aplica los cambios presionando el botón de "aplicar"	29. El sistema valida que no exista ninguna otra tarjeta con ese nombre.
	30. El sistema modifica la descripción de la tarjeta.
	31. El sistema cambia en la lista de tarjetas el nombre de la tarjeta modificada.
	32. El sistema muestra la lista de tarjetas creadas hasta el momento.
Sección: Ayuda	
Acción del actor	**Respuesta del sistema**

33. Al Arquitecto selecciona el botón de "ayuda".	34. El sistema muestra una ventana que ofrece la ayuda de como trabajar con la interfaz de creación del grupo de tarjetas.
35. El arquitecto selecciona el botón "cerrar"	36. El sistema muestra la interfaz para la gestión de las tarjetas.
Flujo Alternativo de Eventos	
Acción del actor	**Respuesta del sistema**
Acción 6	
F1 El Arquitecto presiona el botón “cancelar”	**F1.1** El sistema cierra la ventana para la creación de los grupos de tarjetas, no guarda el grupo creado y muestra nuevamente la interfaz principal de la aplicación.
Acción 20	
F1 El Arquitecto selecciona el botón de “cancelar”	**F1.1** El sistema no elimina la tarjeta seleccionada y muestra la Interfaz de crear tarjetas.
Acción 29	
	F1.0 El sistema verifica que el nombre de la tarjeta a modificarse es el mismo que tenía.
	F1.1 El sistema modifica la descripción de la tarjeta.
	F1.2 El sistema mantiene la lista de tarjetas con los mismo nombres que tenia hasta el momento.
	F1.3 El sistema muestra la lista de tarjetas creadas hasta el momento.
Interfaz: Gestionar grupo de Tarjetas.	

2.7.2.2 Caso de uso: Gestionar el proceso de creación de estudios.

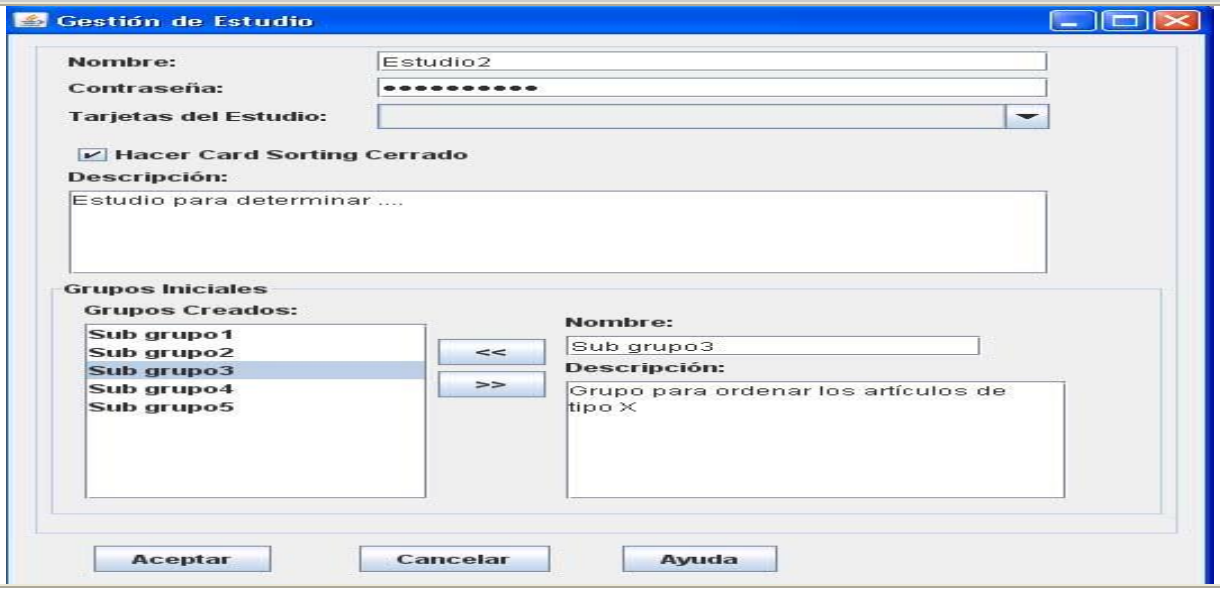
Caso de Uso	
CU-GE	CU_GestionarEstudio
Propósito	El propósito de este Caso de uso es crear el estudio que será publicado para que los usuarios accedan a él.
Actores	Arquitecto
Resumen	El caso de uso se inicia cuando el arquitecto escoge la opción de crear una nuevo grupo de tarjeta y el sistema mostrará una ventana para

	poner nombre y descripción de las mimas, e irá mostrando la lista de las tarjetas que se vayan creando.
Referencias	RF 2
Precondiciones	Tiene que existir al menos un grupo de tarjetas creadas a las cuales se les realizará un estudio.
Pos condiciones	Se crea un estudio listo para ser publicado.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El Arquitecto selecciona en el menú de la interfaz principal la opción de editar, crear nuevo estudio. (ver anexo no. 1)	2. El sistema muestra una ventana "Crear Estudio" para la entrada de datos, como: 1- Un campo para mostrar la lista con los nombres de todos los grupos de tarjetas que hayan sido creados.. 2- Un campo para adicionar el nombre del estudio. 3- Un campo para realizar las descripciones de las tarjetas. 4- Un campo para especificar la contraseña del estudio que se crea. 5- Un campo para la descripción del estudio. 6- Muestra las opciones (botones) de adicionar (flecha <), eliminar (flecha >), aceptar, cancelar y uno para mostrar la ayuda. - Permite modificar el estudio creado.
3. El Arquitecto selecciona la opción que desees (modificar estudio, eliminar estudio o ver ayuda. a) Si se va a modificar. Ver “Sección Modificar”. a) Si selecciona eliminar estudio. Ver "Sección Eliminar". c) Si se quiere ver la ayuda. Ver "Sección Ayuda".	
4. El Arquitecto escribe el nombre del estudio en el campo destinado al nombre.	
5. El Arquitecto en caso de creerlo necesario realiza una descripción del estudio.	
6. El Arquitecto despliega el menú desplegable donde se encuentran todos los grupos	7. El sistema muestra la lista de los grupos de tarjetas creados anteriormente.

de tarjetas creados.	
8. El Arquitecto selecciona el grupo de tarjetas al que le va a realizar el estudio.	
9. El arquitecto marca la opción para realizar la prueba de card Sorting de tipo "cerrado"	10. El sistema habilita la opción para crear las agrupaciones que quedarán como resultado.
11. El arquitecto escribe el nombre y descripción del grupo que esta creando.	
12. El arquitecto presiona la flecha para insertar grupo.	13. El sistema verifica que no haya un grupo creado con el mismo nombre
	14. El sistema adiciona el nombre creado en la lista de los grupos iniciales.
15. El Arquitecto presiona el botón de aceptar.	16. El sistema verifica que no exista otro estudio con ese nombre.
	17. El sistema crea el fichero en la dirección especificada y muestra nuevamente la interfaz principal.
	18. Finaliza el caso de uso.
Sección: Modificar	
Acción del actor	**Respuesta del sistema**
19. El Arquitecto marca en la lista de estudios que muestra la interfaz principal el estudio que se desea modificar.	
20. El Arquitecto selecciona en el menú de la interfaz principal la opción de modificar estudio	21. El sistema muestra la interfaz para gestionar el estudio seleccionado
22. El Arquitecto modifica nombre, contraseña en los campos correspondientes.	
23. El Arquitecto cambia el tipo de Card Sorting que se va a realizar. a) Si es Card Sorting cerrado. Ver "Sección: CS cerrado". b) Si es Card Sorting abierto. Ver "Sección: CS abierto".	
24. El arquitecto aplica los cambios presionando el botón de "aceptar"	25. El sistema valida que no exista ningún otro estudio con ese nombre.
	26. El sistema modifica los cambios realizados en el estudio.
	27. El sistema cambia el nombre del estudio.
	28. El sistema muestra el estudio en la lista de estudios de la interfaz

	principal.
Sección: CS cerrado	
Acción del actor	**Respuesta del sistema**
29. El Arquitecto modifica nombre y la descripción de las agrupaciones iniciales	30. El sistema muestra los cambios en la lista de agrupaciones y en las descripciones de los grupos.
Sección: CS abierto	
Acción del actor	**Respuesta del sistema**
31. El arquitecto desmarca la opción de Card Sorting "cerrado"	
Sección: Eliminar	
Acción del actor	**Respuesta del sistema**
32. El arquitecto selecciona uno de los estudios de la lista	
33. El Arquitecto presiona la opción para "eliminar" que se encuentra en el menú Editar de la interfaz principal	34. El sistema muestra el mensaje "¿Realmente desea eliminar el estudio seleccionado?"
35. El Arquitecto selecciona el botón de aceptar.	36. El sistema elimina el estudio antes seleccionada de la lista de los estudios creada.
	37. El sistema muestra el listado de los estudios modificado (sin el que se eliminó).
Sección: Ayuda	
Acción del actor	**Respuesta del sistema**
38. Al Arquitecto selecciona el botón de "ayuda".	39. El sistema muestra una ventana que ofrece la ayuda de como trabajar con la interfaz de creación de estudios
40. El arquitecto selecciona el botón "cerrar"	41. El sistema muestra la interfaz para la creación de estudios.
Flujo Alternativo de Eventos	
Acción del actor	**Respuesta del sistema**
Acción 9	
F1 El arquitecto desmarca la opción para realizar la prueba de card Sorting de tipo "cerrado"	**F1.1** El sistema des - habilita la opción para crear las agrupaciones que quedarán como resultado.
Acción 15	
F1 El Arquitecto presiona el botón de cancelar	**F1.1** El sistema no guarda el estudio y muestra nuevamente le interfaz principal.
Acción 19	
F1 El Arquitecto NO marca en la lista de estudios que muestra la interfaz principal	

el estudio que se desea modificar.	
F1.1 El Arquitecto selecciona en el menú de la interfaz principal la opción de Modificar estudio.	**F1.2** El sistema muestra el mensaje "Ud. debe seleccionar un Estudio en la lista de estudios."
F1.3 El Arquitecto presiona el botón de aceptar.	**F1.4** El sistema muestra la interfaz principal.
Acción 25	
	F1.0 El sistema verifica que el nombre del estudio a modificarse es el mismo.
	F1.1 El sistema modifica la descripción del estudio
	F1.2 El sistema mantiene la lista de estudios con los mismos nombres que tenía hasta el momento.
	F1.3 El sistema muestra en la interfaz principal la lista de estudios que había hasta el momento.
Acción 35	
F1 El Arquitecto selecciona el botón de “cancelar”	**F1.1** El sistema no elimina el estudio seleccionado de la lista y muestra nuevamente la interfaz principal.
Interfaz: Gestionar Estudio.	



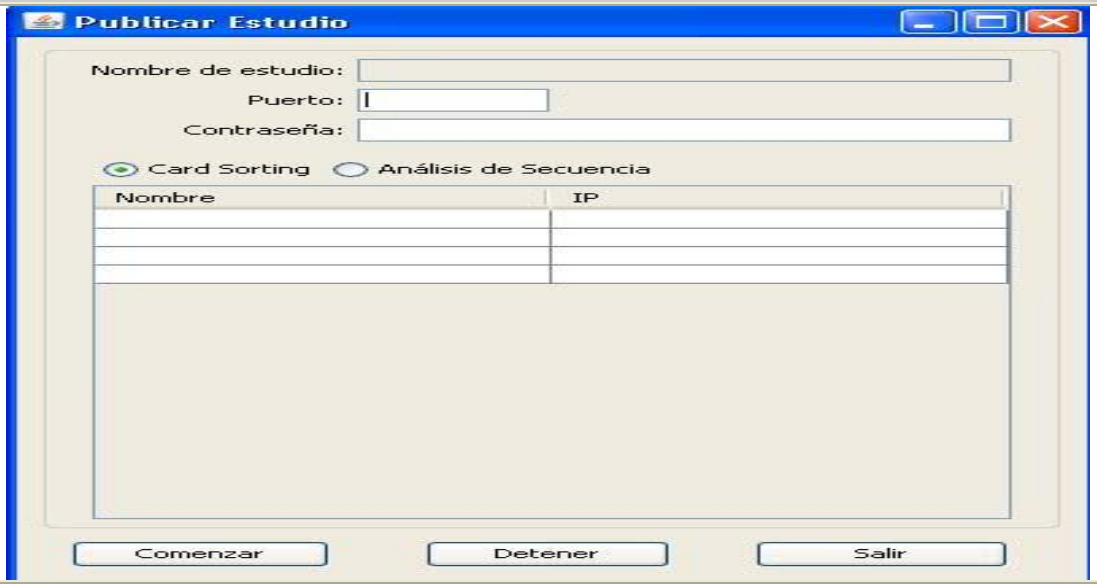
2.7.2.3 Caso de uso: Publicar Estudio.

Caso de uso	
CU-PE	CU_PublicarEstudio
Propósito	Publicar un determinado estudio para que los usuarios se conecten a la aplicación servidora y lo lleven a cabo.
Actores	Arquitecto
Resumen	El caso de uso se inicia cuando el arquitecto selecciona en el menú principal la opción de publicar un estudio y el sistema mostrará una

	ventana para configurar los datos del estudio. Esta ventana estará activa durante todo el tiempo que demore el estudio e irá mostrando la lista de los nombres y direcciones IP de los usuarios que se encuentran realizando el estudio.
Referencias	RF 3
Precondiciones	Debe existir al menos un estudio creado.
Pos condiciones	Se obtiene la realización del estudio por parte de los usuarios.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El Arquitecto selecciona en la lista de estudios de la interfaz principal, el que desea publicar.	
2. El Arquitecto en el menú principal de la aplicación, en el sub - menú "estudio", selecciona la opción de publicar estudio.	3. El sistema muestra una interfaz con el nombre del estudio a publicar y algunos campos para definir datos de la publicación y ver algunos detalles de la conexión. Muestra los campos para establecer el puerto de la conexión y para establecer la contraseña. Brinda la opción para seleccionar que tipo de prueba es la que se realizará. Muestra un panel ´para visualizar cuales son los usuarios que están conectados haciendo el estudio y el Ip de la PC en que se encuentran. Muestra los botones de comenzar, detener y salir.
4. El Arquitecto especifica el puerto por el que se establecerá la conexión	
5. El Arquitecto especifica la contraseña por la que accederá el usuario a realizar el estudio	
6. El Arquitecto selecciona el tipo de prueba que va a aplicar. a) Si es Card Sorting. Ver "Sección Card Sorting" b) Si es Análisis de Secuencia. Ver "Sección Análisis de Secuencia"	
7. El arquitecto selecciona el botón de salir.	8. El sistema verifica que no este corriendo la aplicación servidora.
	9. El sistema cierra la ventana de publicación del estudio y muestra la Interfaz principal.

	10. Finaliza el caso de uso
Sección: Card Sorting	
Acción del actor	**Respuesta del sistema**
11. El Arquitecto selecciona la opción de comenzar la prueba.	12. El sistema publica el estudio en la red
	13. El sistema escucha las peticiones de conexión.
	14. El sistema verifica que los datos de la conexión entrante son correctos.
	15. El sistema envía los mensajes a la aplicación cliente (mensajes de: "HACER_CARG_SORTING", mensaje del tipo de card Sorting que se este realizando " CARD_SORTING_CERRADO" y envía un mensaje además con el grupo de tarjetas para realizar la prueba)
	16. El sistema muestra el nombre y dirección de IP de los usuarios conectados realizando el la prueba.
	17. El sistema recibe las agrupaciones realizadas por cada uno de los usuarios conectados.
	18. El sistema guarda las agrupaciones realizadas por cada usuario.
19. El usuario presiona el botón "detener"	20. El sistema detiene la publicación.
Sección Análisis de Secuencia	
Acción del actor	**Respuesta del sistema**
21. El Arquitecto selecciona la opción de comenzar la prueba.	22. El sistema publica el estudio en la red.
	23. El sistema escucha las peticiones de conexión
	24. El sistema verifica que los datos de la conexión entrante son correctos.
	25. El sistema envía los mensajes a la aplicación cliente (mensajes de: "HACER_ANALISIS_SECUENCIA," y envía un mensaje además con la agrupación resultante luego de haber aplicado el algoritmo)
	26. El sistema muestra el nombre y dirección de IP de los usuarios conectados realizando el la prueba.
	27. El sistema recibe los ordenamientos realizados por cada uno de los usuarios conectados.
	28. El sistema guarda los ordenamientos realizados por cada usuario
29. El usuario presiona el botón "detener"	30. El sistema detiene la publicación.
Flujo Alternativo de Eventos	
Acción del actor	**Respuesta del sistema**
Acción 1	
F1 El Arquitecto NO selecciona en la lista de estudios de la interfaz principal, el que desea publicar.	
F1.1 El Arquitecto en el	**F1.2** El sistema muestra el mensaje "Ud. debe seleccionar un estudio

menú principal de la aplicación, en el sub - menú "estudio", selecciona la opción de publicar estudio.	de la lista."
F1.3 El Arquitecto presiona el botón "aceptar"	**F1.4** El sistema muestra la lista de estudios creados.
Acción 8	
	F1.0 El sistema verifica que la aplicación servidora aún esta corriendo
	F1.1 El sistema muestra el mensaje: "Prueba en ejecución, presione el botón para detener la prueba"
F1.2 El Arquitecto presiona el botón "aceptar"	**F1.3** El sistema muestra la interfaz de publicar el estudio.
F1.4 El arquitecto presiona el botón "detener"	
F1.5 El arquitecto presiona el botón "salir"	**F1.6** El sistema cierra la ventana de publicación del estudio y muestra la interfaz principal de la aplicación.
Acción 14	
	F1.0 El sistema verifica que los datos de conexión son incorrectos.
	F1.1 El sistema envía mensajes a los usuarios conectados cuando hay problemas con la conexión como: "Contraseña incorrecta.", "Ya existe un usuario conectado desde ese IP.", "Datos no válidos."
Acción 24	
	F1.0 (Se repite el flujo alternativo de la *Acción 15)*
Interfaz: PublicarEstudio.	



2.7.2.4 Caso de uso: Establecer Conexión.

Caso de uso	
CU-EC	CU_Establecer_Conexión
Propósito	Conectarse a la PC servidora y poder acceder y realizar el estudio que

	haya sido publicado por el arquitecto.
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario ejecuta la aplicación cliente y el sistema muestra una ventana para configurar los datos del estudio al cual se va a acceder. Mediante esta interfaz el usuario se conectará a la aplicación servidora y obtendrá los datos del estudio determinado que haya sido publicado.
Referencias	RF 4
Precondiciones	Debe haberse publicado un estudio para que se establezca una conexión
Pos condiciones	La aplicación cliente guarda los datos de todos los usuarios que han realizado el estudio.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario ejecuta la aplicación cliente.	2. El sistema muestra la interfaz de la aplicación cliente con los datos de conexión tales como: - Un campo para identificar el usuario que realizará la prueba. - Un campo para especificar la contraseña de acceso al estudio publicado. - Un campo para establece el IP de la PC servidora - Un campo para especificar el puerto por el que se establecerá la conexión. Muestra el nombre del estudio al que se esta accediendo así como su descripción. Muestra los botones de conectar, desconectar, enviar y terminar.
3. El usuario escribe su nombre en el campo para establecer el nombre del usuario.	
4. El usuario inserta la dirección de la PC servidora en el campo "IP"	
5. El usuario Especifica el puerto de la conexión el campo "Puerto"	
6. El usuario introduce la contraseña de acceso en el campo "contraseña".	
7. El usuario presiona el botón de "conectar"	8. El sistema comienza la conexión con la aplicación servidora.
	9. El sistema muestra el nombre y la descripción del estudio al que se tiene acceso.
	10. EL sistema muestra una interfaz desplegable para realizar el tipo de estudio publicado por el servidor.

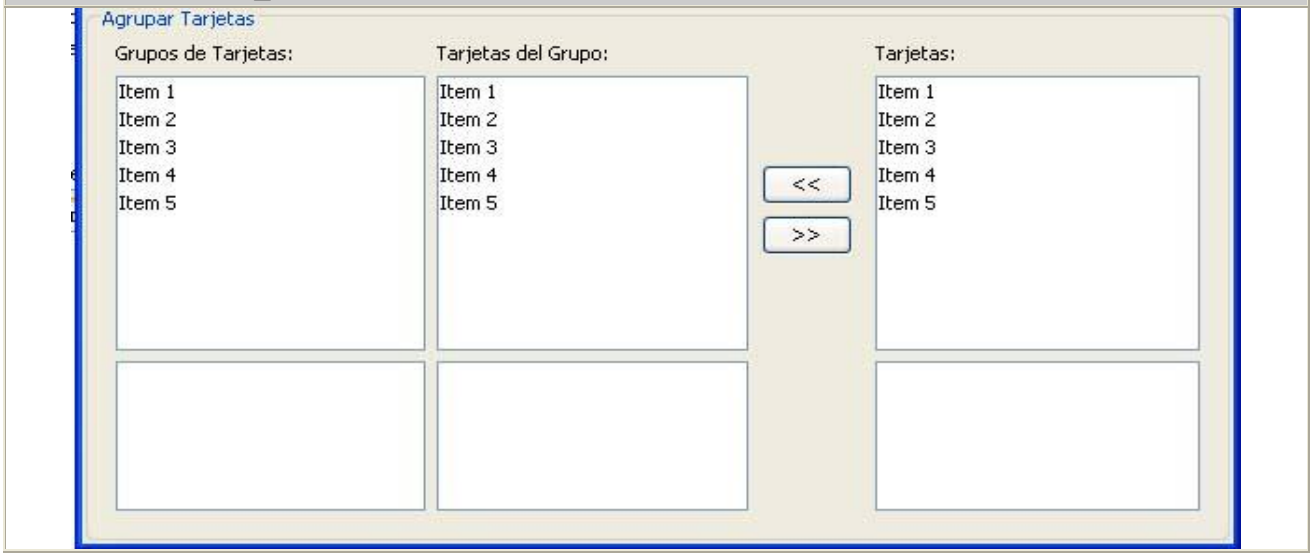
11. El usuario realiza el estudio al que tuvo acceso.	
12. El usuario presiona el botón "enviar".	13. El sistema envía un mensaje con los datos del estudio realizado a la aplicación servidora.
14. El usuario presiona el botón "detener".	15. El sistema detiene la conexión.
	16. El sistema habilita el botón de "terminar"
17. El usuario presiona el botón "terminar"	18. El sistema cierra la interfaz de la aplicación cliente
	19. Finaliza el caso de uso.
Flujo Alternativo de Eventos	
Acción del actor	**Respuesta del sistema**
Acción 8	
	F1.0 El sistema no establece conexión con la aplicación servidora.
	F1.1 El sistema muestra un mensaje de "error en la conexión" o "error en la entrada de los datos"
	F1.2 El sistema muestra la interfaz de la aplicación cliente de establecer conexión y pasa a la acción No. 4
Acción 12	
F1 El usuario no envía el estudio realizado.	
F1.1 El usuario presiona el botón "detener".	**F1.2** El sistema no envía a la aplicación servidora los datos del estudio
	F1.3 El sistema detiene la conexión.

2.7.2.5 Caso de uso: Realizar Agrupaciones.

Caso de uso	
CU-RA	CU_Realizar_Agrupaciones
Propósito	Repartir las tarjetas de un determinado estudio en los diferentes grupos creados.
Actores	Usuario
Resumen	Este caso de uso comienza una vez que estén creadas todas las tarjetas con las que se realizará el estudio y se escoge la opción de realizar agrupaciones. El sistema muestra una ventana con el listado de las tarjetas y se podrán ir creando los grupos con el nombre deseado y repartiendo las tarjetas en dichos grupos. El sistema también mostrará los listados de estos grupos y el contenido de los mismos.
Referencias	RF 5
Precondiciones	Tiene que haberse establecido una conexión previa con la aplicación servidora.
Pos condiciones	Enviar a la aplicación servidora las agrupaciones de tarjetas realizadas por todos los usuarios.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
	1. El sistema despliega una ventana con los campos para mostrar los grupos creados en el estudio, así como para ver la descripción de los

	mismos. Muestra otro campo con el grupo de tarjetas creado para el estudio. Y los botones para adicionar y sacar tarjetas de un grupo determinado. (botón "<<" y botón >>)
2. El usuario selecciona un grupo en la lista de grupos predeterminados.	3. El sistema muestra la descripción del grupo seleccionado.
4. El usuario selecciona la opción que desee. Si va a adicionar una tarjeta a un grupo. Ver "Sección Adicionar tarjeta a un grupo " (botón<<) Si va a quitar una tarjeta de un grupo. Ver "Sección Sacar tarjeta de un grupo " (botón>>)	
5. El usuario selecciona el botón "enviar"	6. El sistema verifica que las tarjetas hayan sido repartidas en los grupos.
	7. Termina el caso de uso.
Sección: Adicionar tarjeta a Grupo	
Acción del actor	**Respuesta del sistema**
8. El usuario selecciona en la lista de tarjetas una de las tarjetas.	9. El sistema muestra la descripción de la tarjeta.
10. El usuario presiona el botón para adicionar la tarjeta (botón: <<)	11. El sistema elimina la tarjeta de la lista de tarjetas.
	12. El sistema adiciona la tarjeta a la lista de tarjetas perteneciente al grupo seleccionado.
Sección: Sacar tarjeta de un grupo	
Acción del actor	**Respuesta del sistema**
13. El usuario selecciona en la lista de tarjetas del grupo seleccionado una de las tarjetas.	14. El sistema muestra la descripción de la tarjeta.
15. El usuario presiona el botón para adicionar la tarjeta (botón: >>)	16. El sistema elimina la tarjeta de la lista de tarjetas del grupo seleccionado.
	17. El sistema adiciona la tarjeta a la lista de tarjetas general, donde podrá volver a ser seleccionada para pertenecer a otro grupo.
Flujo Alternativo de Eventos	
Acción del actor	**Respuesta del sistema**

Acción 6	
	F1.0 El sistema detecta tarjetas sin repartir
	F1.1 El sistema muestra el mensaje "Aún quedan tarjetas sin ser repartidas."
F1.2 El usuario selecciona el botón "ok"	**F1.3** El sistema muestra la ventana de realizar agrupaciones y pasa a la acción 5.
Interfaz: Realizar_Ordenamientos.	



2.7.2.6 Caso de uso: Realizar Ordenamientos.

Caso de uso	
CU-RO	Cu_Realizar_Ordenamientos
Propósito	Establecer el orden de las agrupaciones resultantes de un determinado estudio
Actores	Usuario
Resumen	El caso de uso se inicia una vez que estén creadas todas las agrupaciones (o sea, todos los grupos de artículos). El sistema muestra una ventana con todas las agrupaciones de determinado estudio y el arquitecto procede a darle orden a dichas agrupaciones según lo hayan realizado los diferentes usuarios que realizaron el estudio, permitiendo guardar cada uno de los ordenamientos.
Referencias	RF 6
Precondiciones	Debe haberse establecido una conexión previamente con la aplicación servidora.
Pos condiciones	Se obtienen las diferentes agrupaciones ordenadas por todos los usuarios que han realizado el estudio.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
	1. El sistema despliega una ventana con los campos para

	mostrar los grupos devueltos por en Card Sorting así como para ver la descripción de los mismos. Muestra también los botones de "subir" y "bajar".
2. El usuario selecciona una agrupación de la lista.	3. El sistema muestra la descripción de la agrupación.
4. El usuario selecciona la opción que desea realizar. Si es subir. Ver "sección Subir". Si es bajar. Ver "sección Bajar".	
5. El usuario selecciona el botón "enviar"	6. El sistema envía los datos de los ordenamientos a la aplicación servidora.
	7. El sistema cierra la ventana de realizar los ordenamientos.
	8. Termina el caso de uso.
Sección: Subir	
Acción del actor	**Respuesta del sistema**
	9. El sistema mueve la agrupación seleccionada un lugar hacia arriba en la lista de agrupaciones.
Sección: Bajar.	
Acción del actor	**Respuesta del sistema**
	10. El sistema mueve la agrupación seleccionada un lugar hacia abajo en la lista.
Flujo Alternativo de Eventos	
Acción del actor	**Respuesta del sistema**
Interfaz: Realizar_Ordenamientos.	

2.7.2.7 Caso de uso: Realizar Análisis Co-ocurrencias.

Caso de uso	
CU-RAC	CU_RealizarA_Coocurrencias
Propósito	Mostrar los resultados de haber aplicado Card Sorting y Análisis de Secuencia.
Actores	Arquitecto.
Resumen	Este caso de uso se inicia una vez que se hayan realizado las agrupaciones de artículos o el ordenamiento de los grupos de artículos, cuando el arquitecto escoge la acción de realizar Card Sorting o la acción de realizar Análisis de Secuencia. El sistema muestra dependiendo del tipo de prueba que se haya realizado.
Referencias	RF 7 y RF 8
Precondiciones	El usuario debe haber realizado los ordenamientos y las agrupaciones de tarjetas. Deben estar los datos del estudio guardados en la PC servidora.
Pos condiciones	Se obtendrán los resultados de las técnicas de haber aplicado Card Sorting y Análisis de Secuencia.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El arquitecto selecciona un estudio para obtener los resultados del mismo	2. El sistema muestra la descripción del estudio.
3. El arquitecto selecciona en el menú de la aplicación servidora la opción para mostrar los resultados	4. El sistema muestra un sub-menú de operaciones como son: "Card Sorting" y "Análisis de Secuencia".
5. El usuario selecciona cual de los dos resultados desea obtener. Si es resultados de Card Sorting. Ver "Sección: Card Sorting". Si es resultados de Análisis de Secuencia. Ver "Sección: Análisis de Secuencia. "	
	6. Finaliza el caso de uso.
Sección: Card Sorting	
Acción del actor	**Respuesta del sistema**
	7. El sistema muestra una ventana con los campos NO editables para mostrar los datos del nombre y descripción del estudio seleccionado. Muestra un campo con los nombres de los grupos y las tarjetas existentes en cada uno de ellos. Que serán las agrupaciones resultantes.

	Luego de haber realizado el algoritmo. (Card Sorting) Muestra los botones de "Realizar Análisis", "Blanquear" y "cerrar"
8. El arquitecto selecciona el botón de "resultados"	9. El sistema comienza a realizar el algoritmo de Card Sorting o (Card Cluster)
	10. El sistema muestra una barra con el avance del proceso del algoritmo.
	11. El sistema en al campo para obtener los resultados muestra los grupos creados y la lista de tarjetas pertenecientes a cada uno de ellos una vez culminado el algoritmo.
12. El usuario selecciona el botón de cerrar la ventana de los resultados botón "cerrar".	13. El sistema cierra la ventana referente a los resultados de Card Sorting
Sección: Análisis de Secuencia	
Acción del actor	**Respuesta del sistema**
	14. El sistema muestra una ventana con los campos NO editables para mostrar los datos del nombre y descripción del estudio seleccionado. Muestra un campo con los nombres de los grupos de tarjetas enumerados en orden ascendente. Que será el ordenamiento resultante. Luego de haber realizado el algoritmo. (Análisis de Secuencia) Muestra los botones de "Realizar Análisis" y "Blanquear" y "cerrar"
15. El arquitecto selecciona el botón de "resultados"	16. El sistema comienza a realizar el algoritmo de Análisis de Secuencia.
	17. El sistema muestra una barra con el avance del proceso del algoritmo.
	18. El sistema en al campo para obtener los resultados muestra la lista ordenada de los grupos resultante una vez culminado el algoritmo.
19. El usuario selecciona el botón de cerrar la ventana de los resultados botón "cerrar".	20. El sistema cierra la ventana referente a los resultados de Análisis de Secuencia.
Flujo Alternativo de Eventos	
Acción del actor	**Respuesta del sistema**
Acción 1	
F1 El Arquitecto No selecciona ningún estudio.	**F1.1** El sistema muestra un mensaje "UD debe seleccionar un estudio".
	F1.2 El sistema muestra la ventana para seleccionar un estudio específico.
Acción 8	
F1 El arquitecto selecciona el botón de "Blanquear"	**F1.1** El sistema elimina los resultados del estudio seleccionado.
	F1.2 El sistema sobre escribe el archivo del estudio en la PC servidora.
Acción 15	
F1 El arquitecto selecciona el botón "Blanquear"	**F1.1** (Se repite el flujo alternativo de la Acción 8)

2.8 Conclusiones

En este capítulo se analizó la propuesta de solución, obteniéndose una lista de las funcionalidades que debe tener el sistema. Se realiza una propuesta del sistema que abarca todo lo que se pretende lograr con la realización del mismo. Las funcionalidades obtenidas fueron representadas mediante un Diagrama de Casos de Uso, y fueron descritas mediante a la descripción de extendida de estos casos de usos. A partir de este punto se puede comenzar a construir el sistema, cumpliendo con todos los requerimientos y las funcionalidades que se determinaron a lo largo del desarrollo de este capítulo.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

Este capítulo tratará el flujo de trabajo Análisis y Diseño explicando como se cumplen los principales objetivos de este flujo de trabajo. Los requisitos se traducen a una especificación que describirá cómo implementar el sistema. Expone además como adaptar el diseño para que sea consistente con el entorno de implementación, diseñado para el rendimiento. Se obtiene además un modelo de clases del análisis por cada caso de uso significativo, y partiendo de los requisitos no funcionales, se realizará el diagrama de clases del diseño así como los diagramas de interacción respectivamente.

3.2 Análisis del sistema

El principal objetivo de la etapa de análisis es entender el problema con mayor claridad, para lograr esto se realizan los diagramas de clases del análisis en los cuales se identifican las clases que describirán las relaciones en los casos de uso. En el análisis se identifican tres tipos de clases, clase entidad, interfaz y control. La primera de estas son clases que modelan información que por lo general poseen larga vida, la segunda modelan la interacción de los actores con el sistema y la última se encarga de la coordinación y realización del caso de uso en específico. A continuación se realizarán los diagramas de clases del análisis correspondiente para cada caso de uso del sistema.

3.2.1 Diagrama de clases del análisis

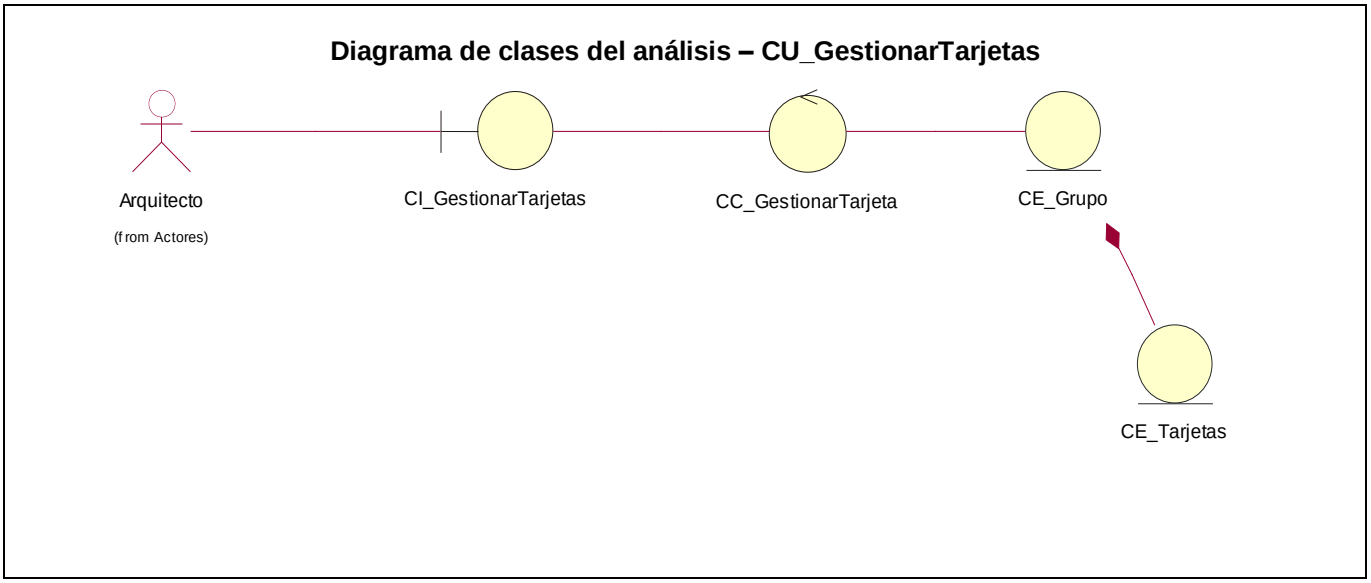


Diagrama de clases del análisis – CU_GestionarEstudio

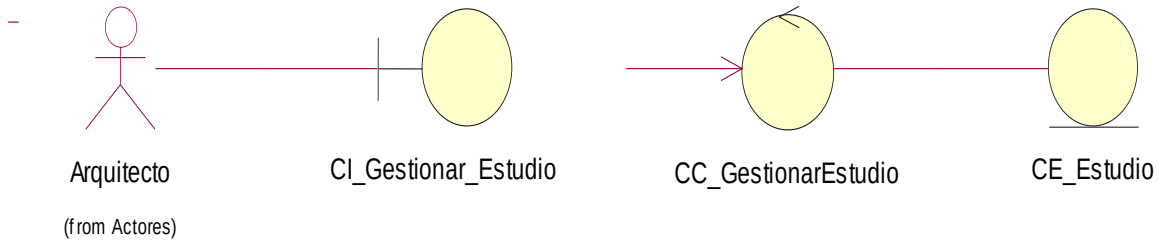


Diagrama de clases del análisis – CU_PublicarEstudio

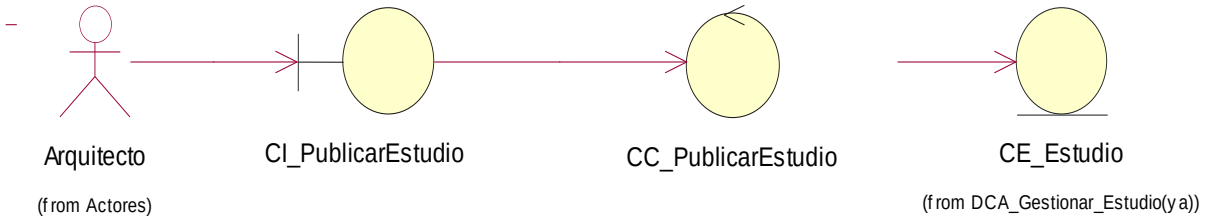


Diagrama de clases del análisis – CU_Establecer_Conexión

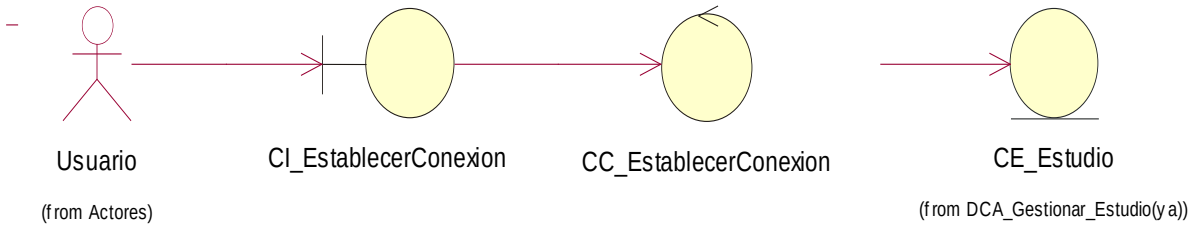


Diagrama de clases del análisis – CU_Gestionar_Agrupaciones

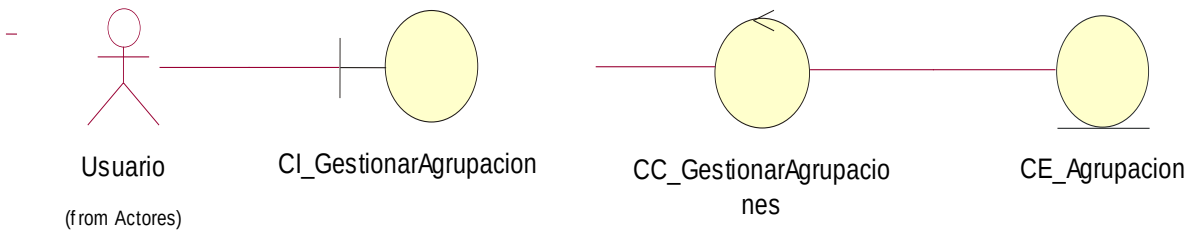


Diagrama de clases del análisis – Cu_Realizar_Ordenamientos

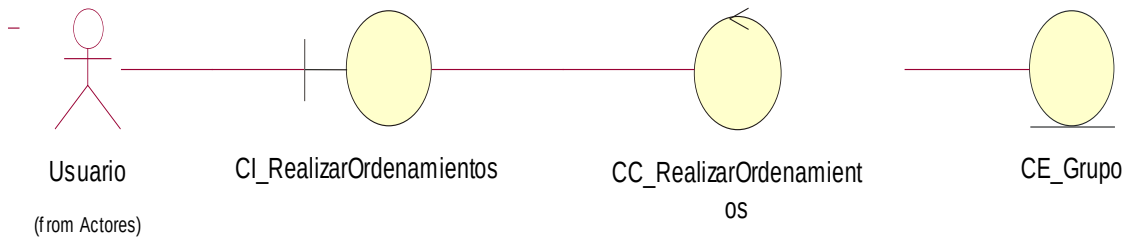
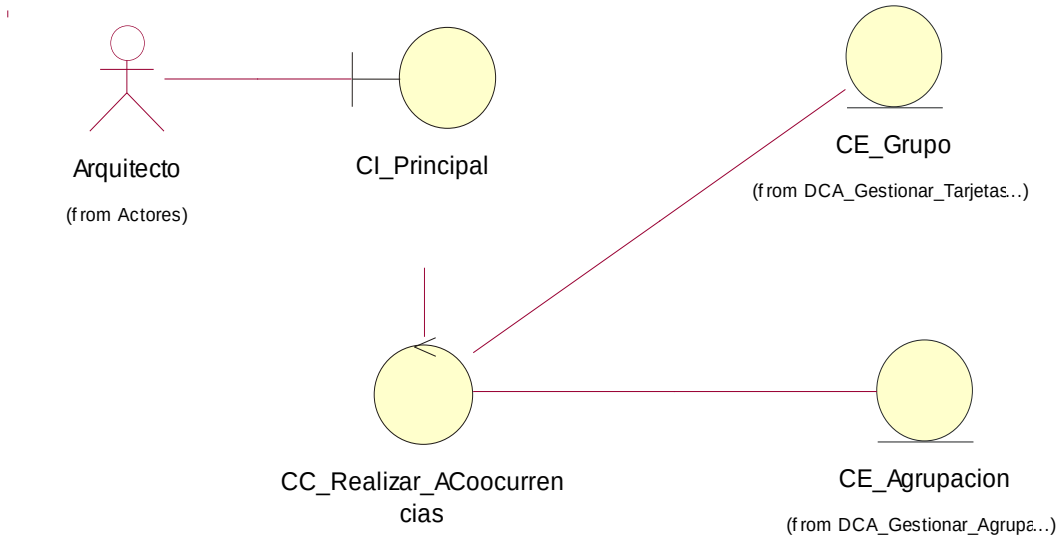


Diagrama de clases del análisis – CU_RealizarA_Coocurrencias



3.3 Diseño del sistema

En el diseño se tiene el propósito de formular los modelos para preparar la entrada a las actividades de implementación y pruebas del sistema, preparando un plano para los artefactos que se crean durante cada uno de estos flujos de trabajo. Se debe definir una solución que satisfaga de modo efectivo y eficiente los requisitos especificados en el análisis, incorpora nuevos artefactos (nuevas clases, nuevos atributos y operaciones para las clases, etc.)

3.3.1 Diagramas de secuencias del diseño.

Los diagramas de secuencia para un determinado caso de uso muestran una interacción ordenada según la secuencia temporal de eventos de un determinado escenario; o sea muestran los objetos participantes en la

interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. A continuación se realizará la descripción de los diagramas de secuencia para cada caso de uso.

DIAGRAMA DE INTERACCION- CU_Gestionar tarjetas (sesión crear)

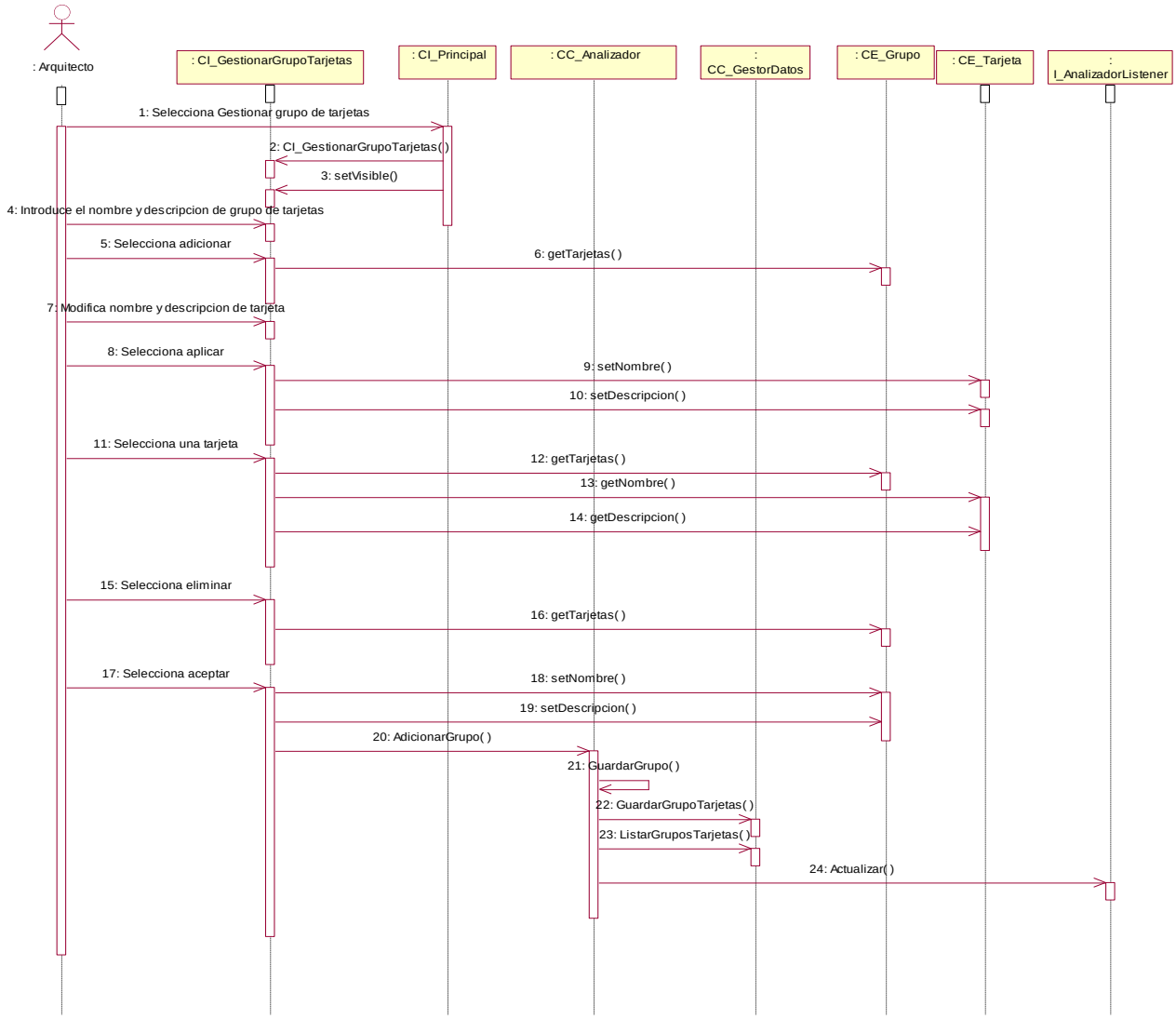


DIAGRAMA DE INTERACCION- CU_Establecer_Conexión

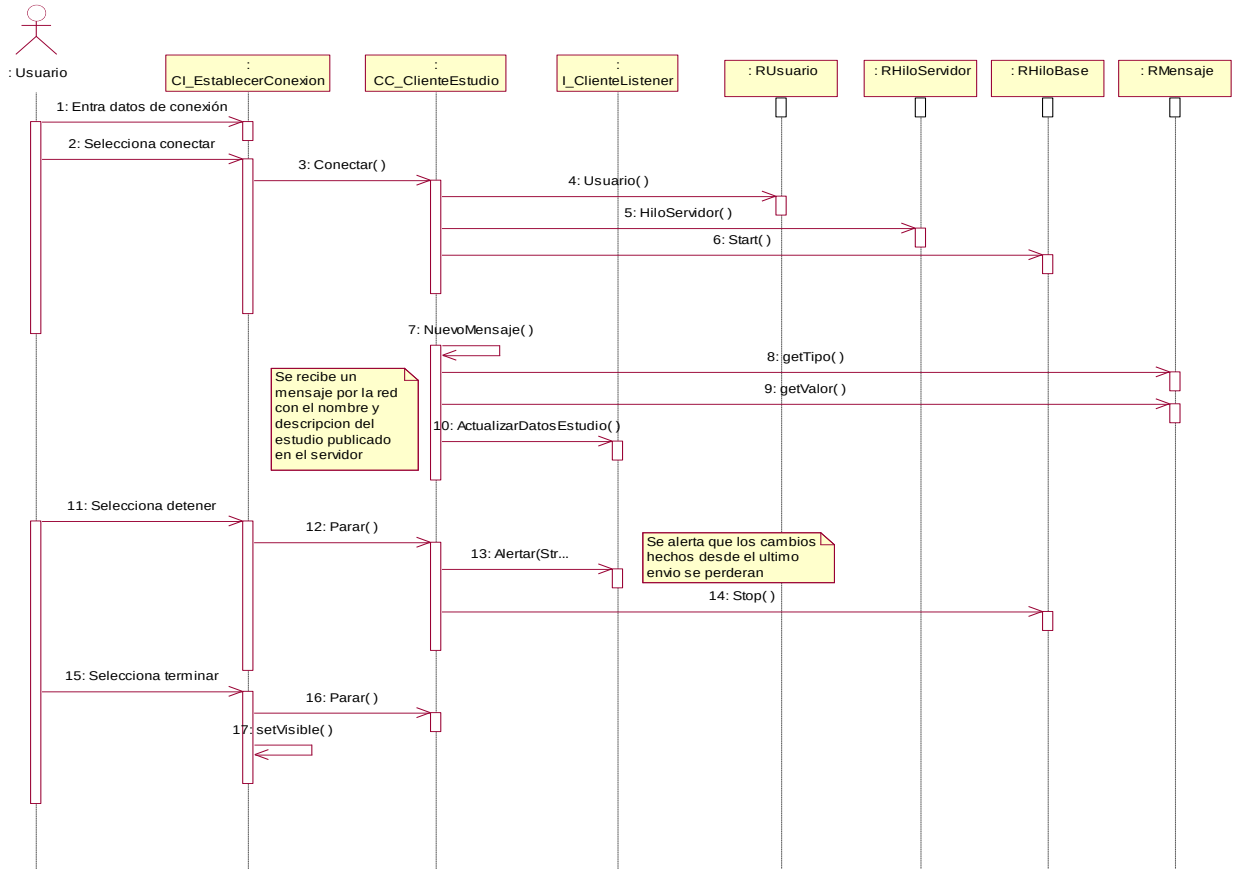


DIAGRAMA DE INTERACCION- CU_Realizar_Agrupaciones

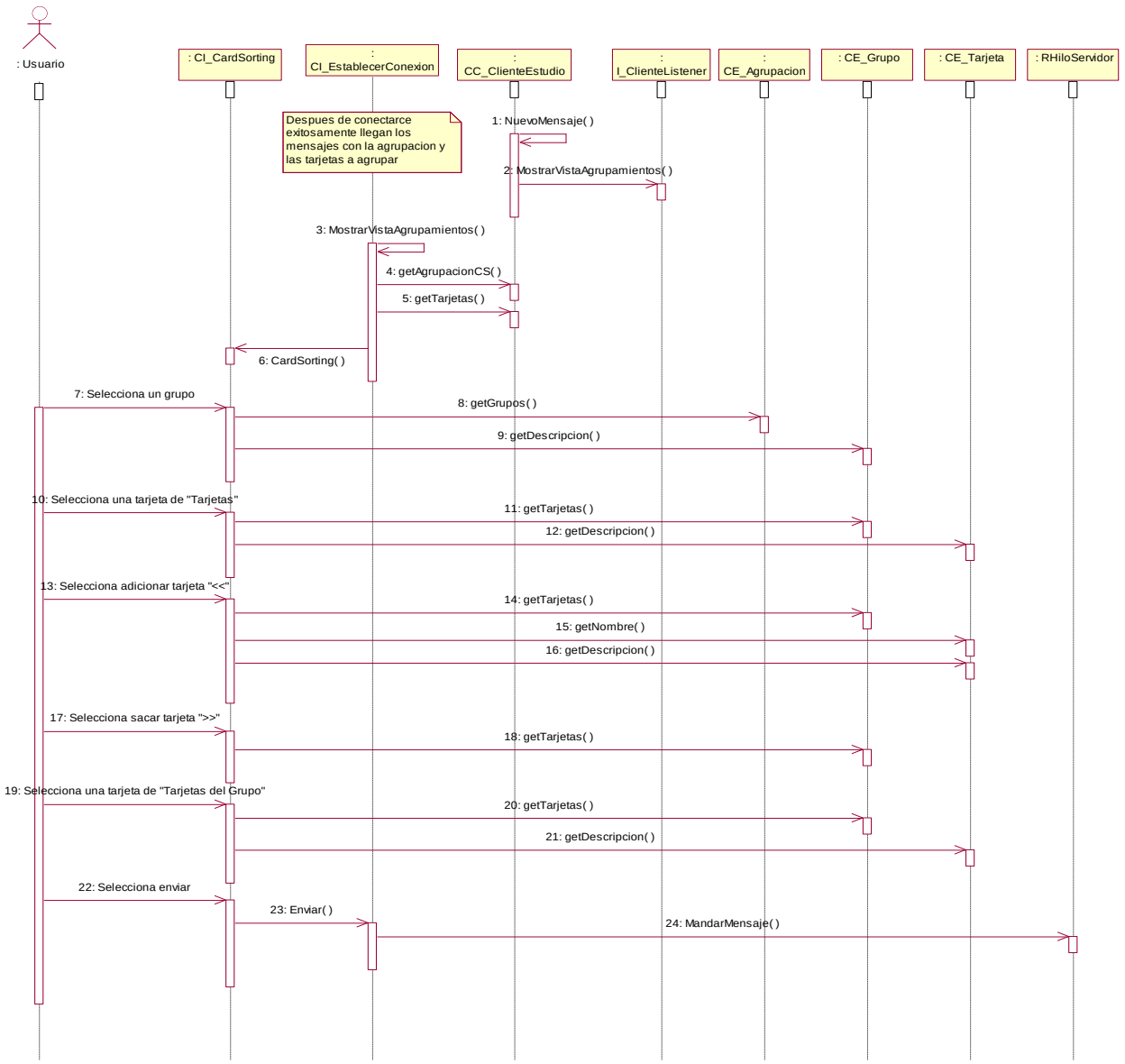


DIAGRAMA DE INTERACCION- Cu_Realizar_Ordenamientos

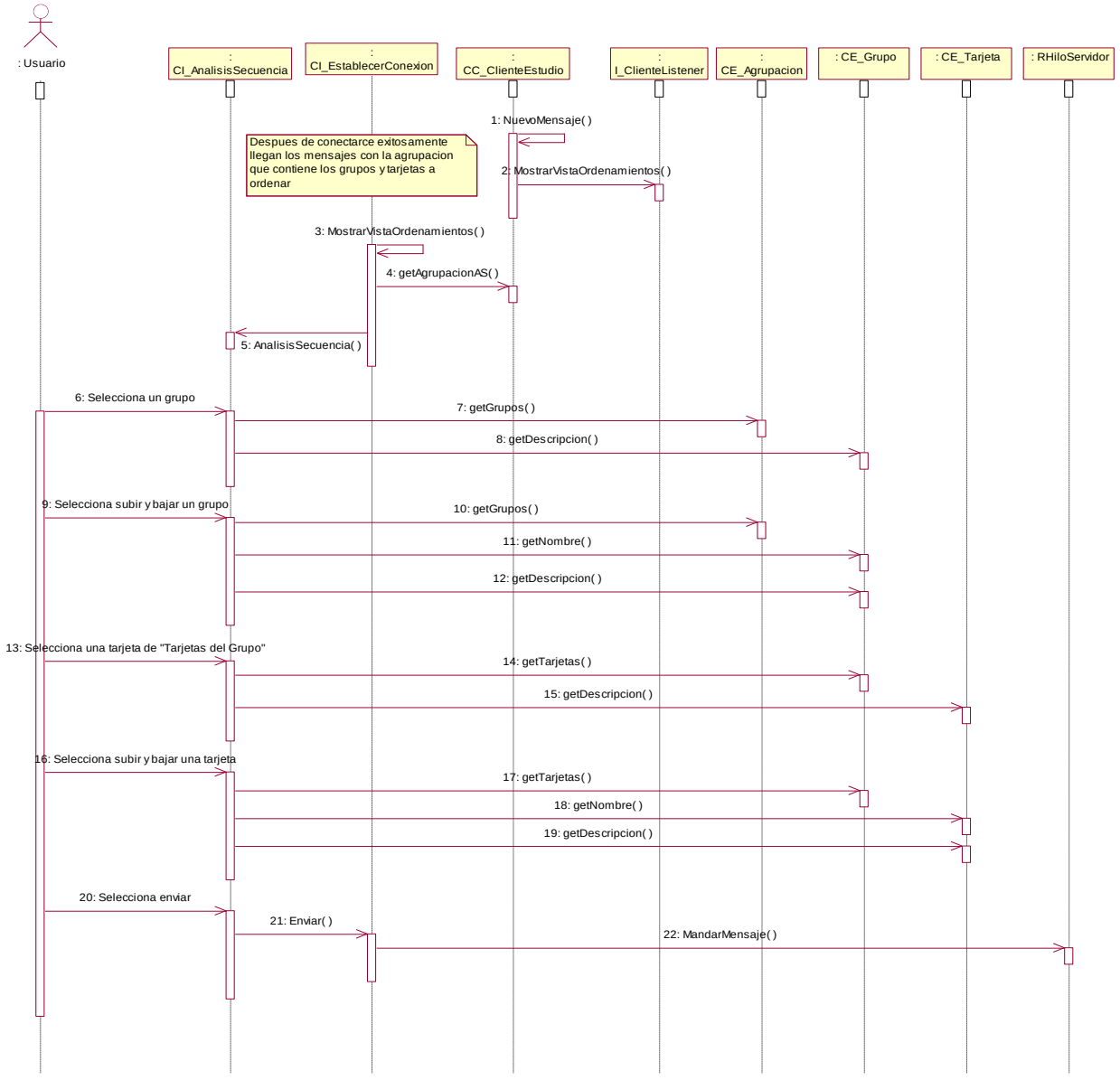
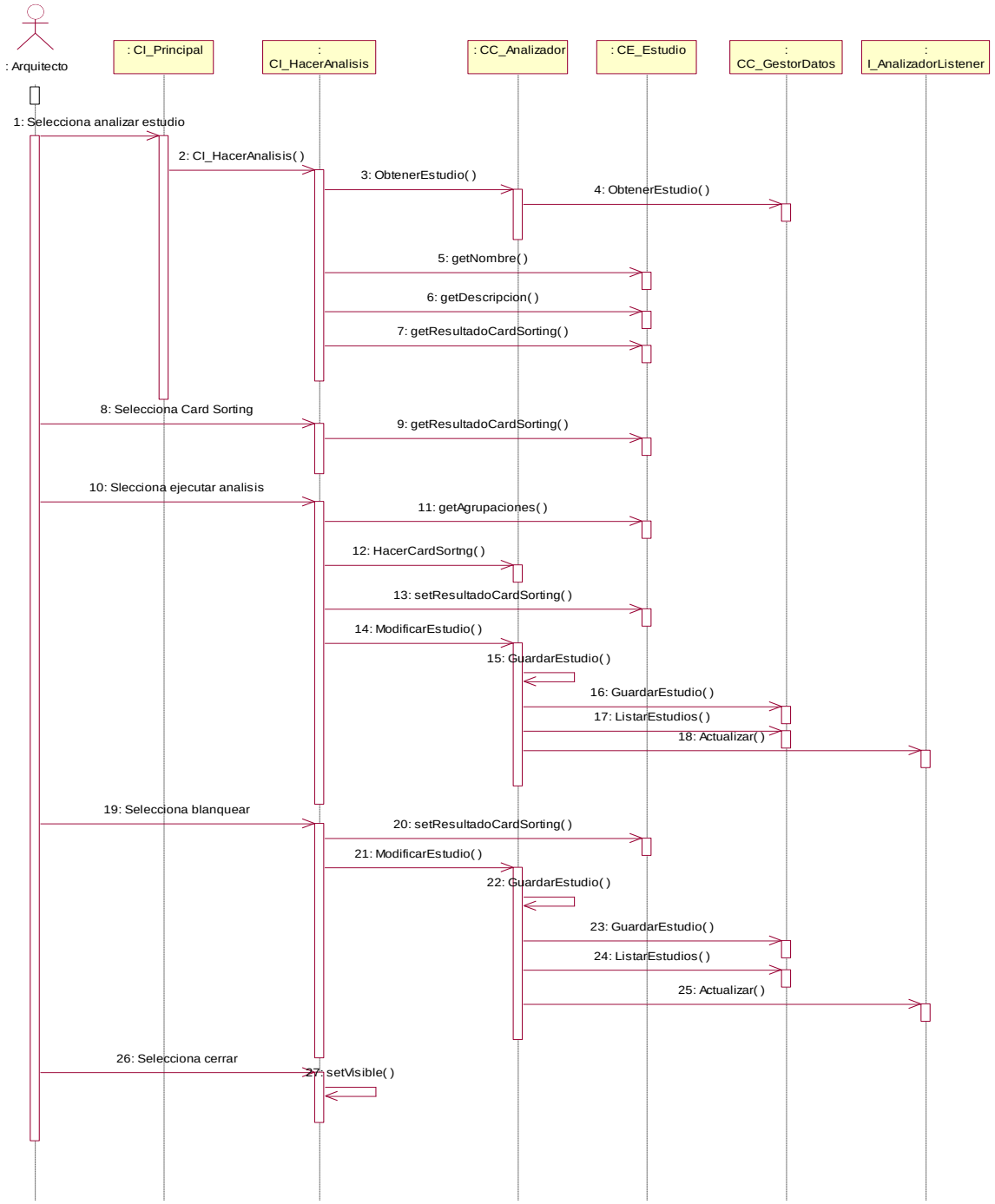


DIAGRAMA DE INTERACCION- CU_RealizarA_Coocurrencias (sesión tipo Card Sorting)



3.3.2 Diagrama de clases del diseño.

Los diagramas de clases son el diagrama principal para el flujo de trabajo análisis y diseño, se basan fundamentalmente en los diagramas de interacción. Presentan las clases del sistema con sus relaciones estructurales y de herencia. Dentro de las partes que conforman un diagrama de clases se encuentran: clases, relaciones y paquetes.

DIAGRAMA DE CLASES DEL DISEÑO – CLASES PERSISTENTES

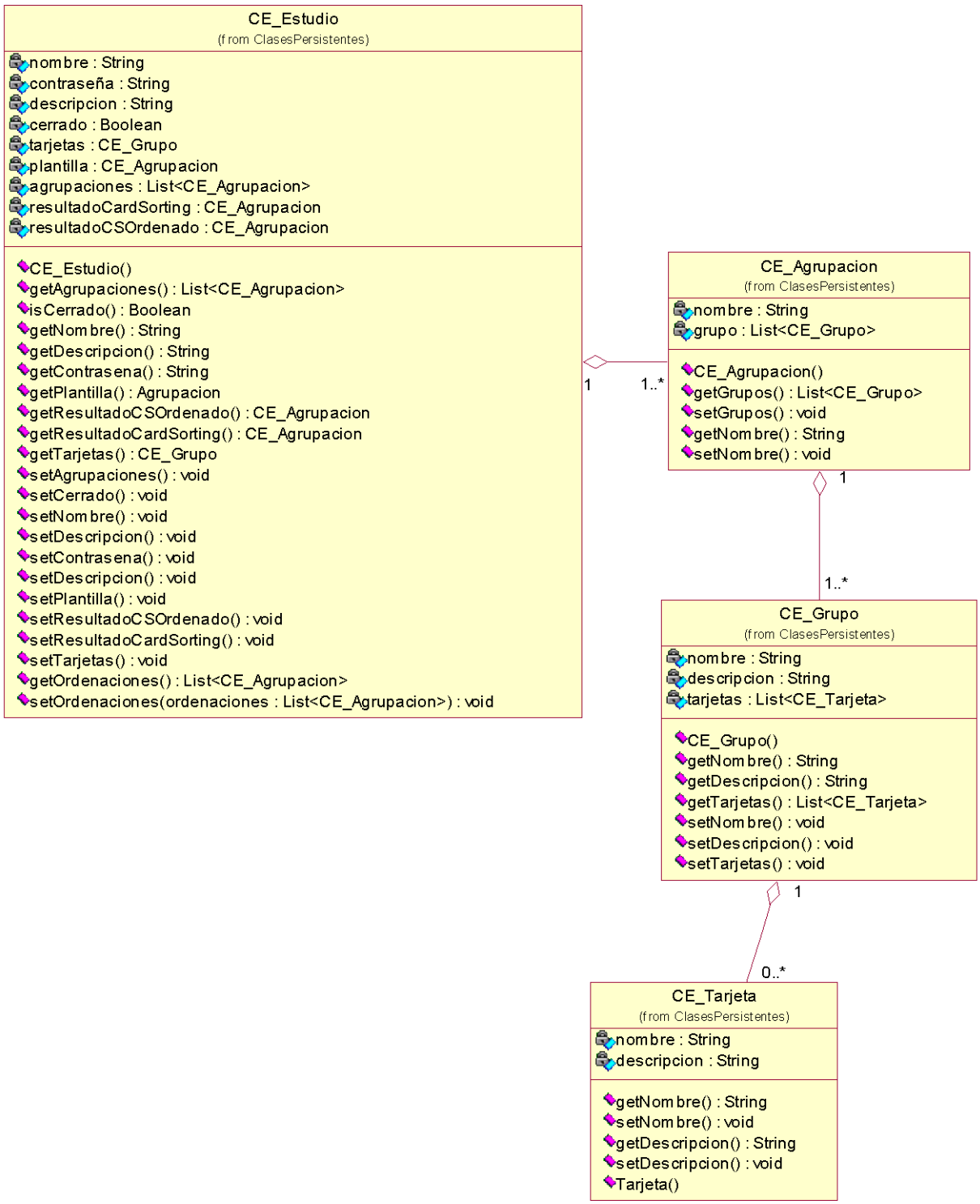


DIAGRAMA DE CLASES DEL DISEÑO – CU_GestionarTarjetas

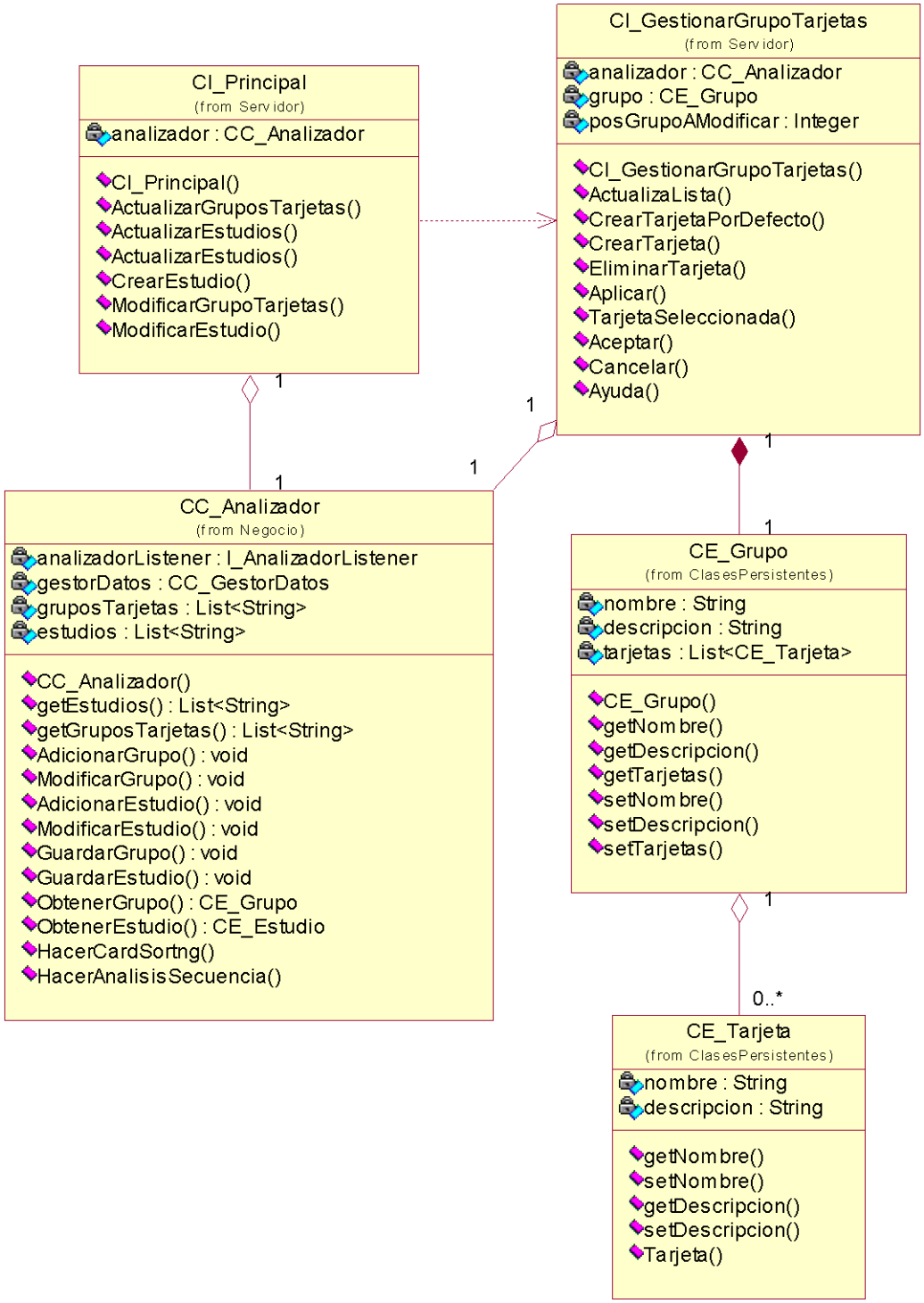


DIAGRAMA DE CLASES DEL DISEÑO – CU_GestionarEstudio

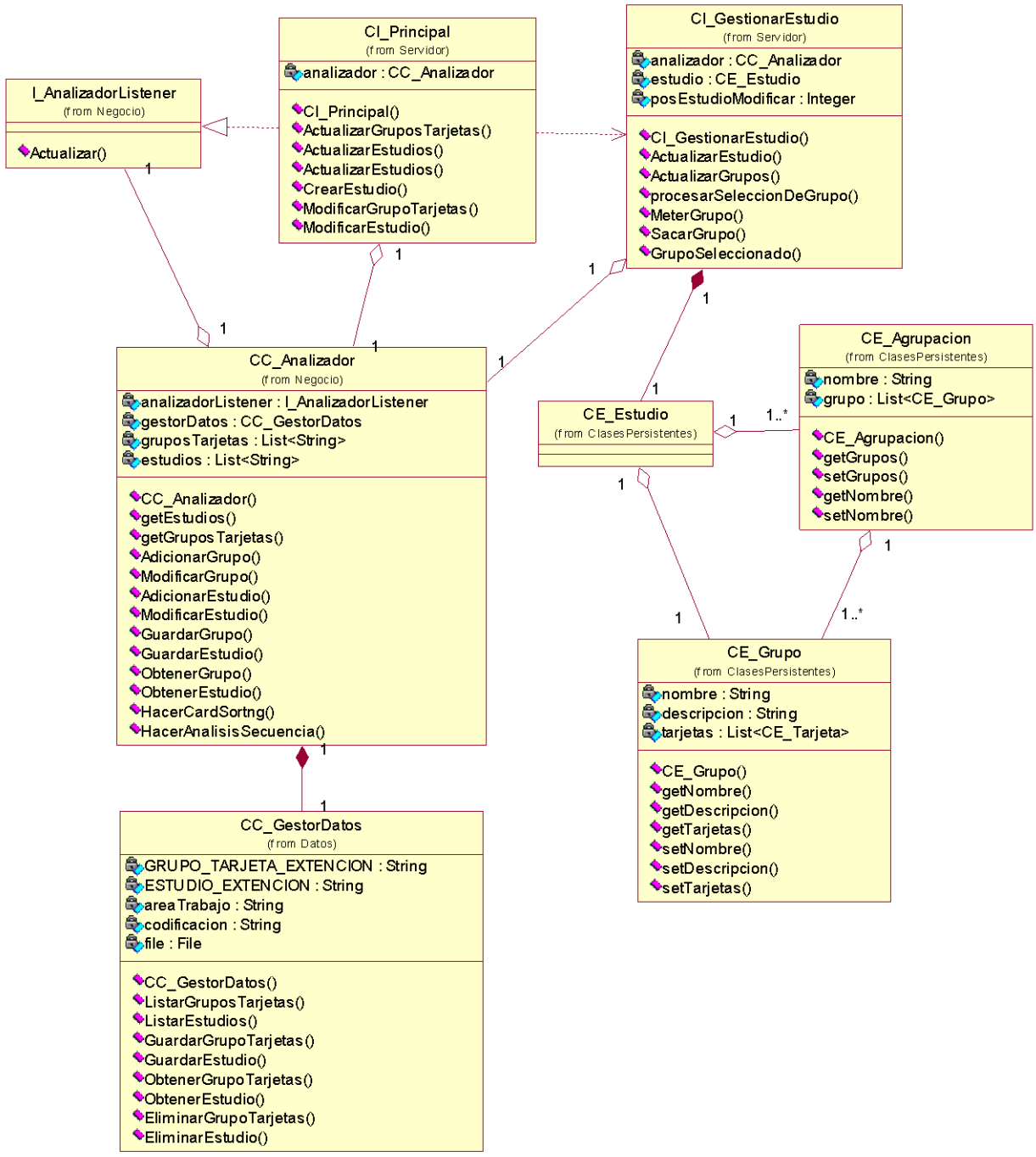


DIAGRAMA DE CLASES DEL DISEÑO – CU_PublicarEstudio

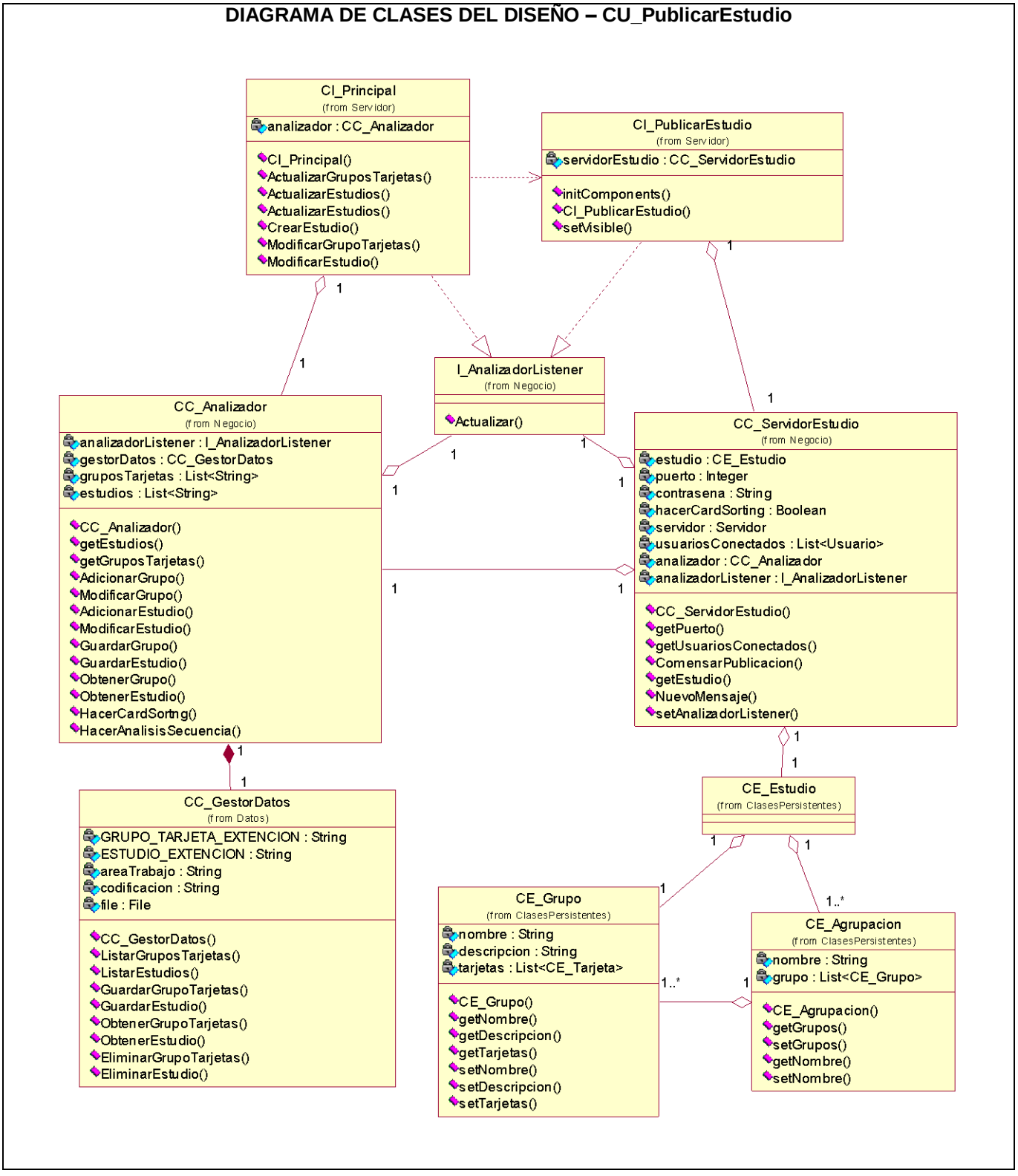


DIAGRAMA DE CLASES DEL DISEÑO – CU_Establecer_Conexión

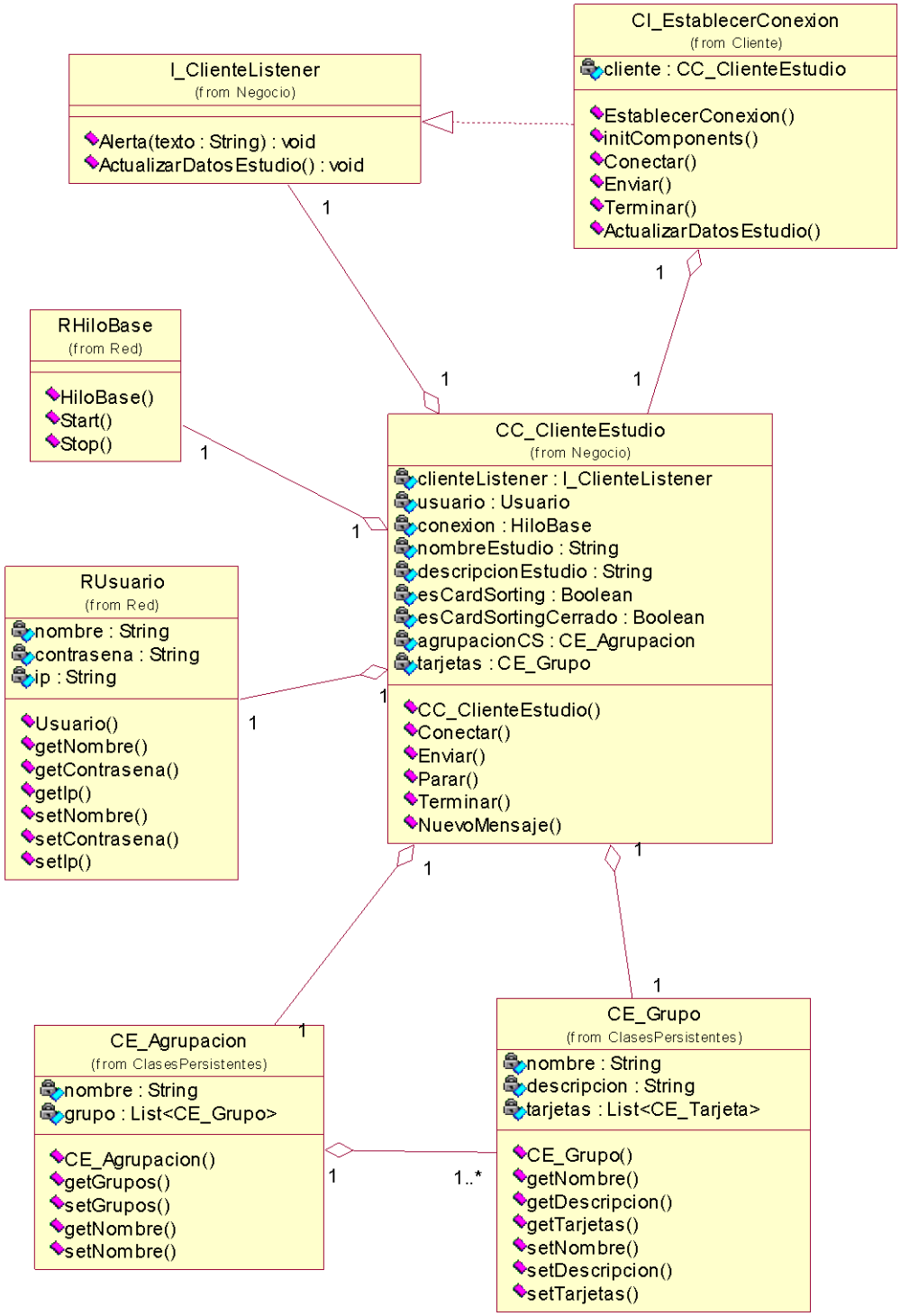


DIAGRAMA DE CLASES DEL DISEÑO – CU_Realizar_Agrupaciones

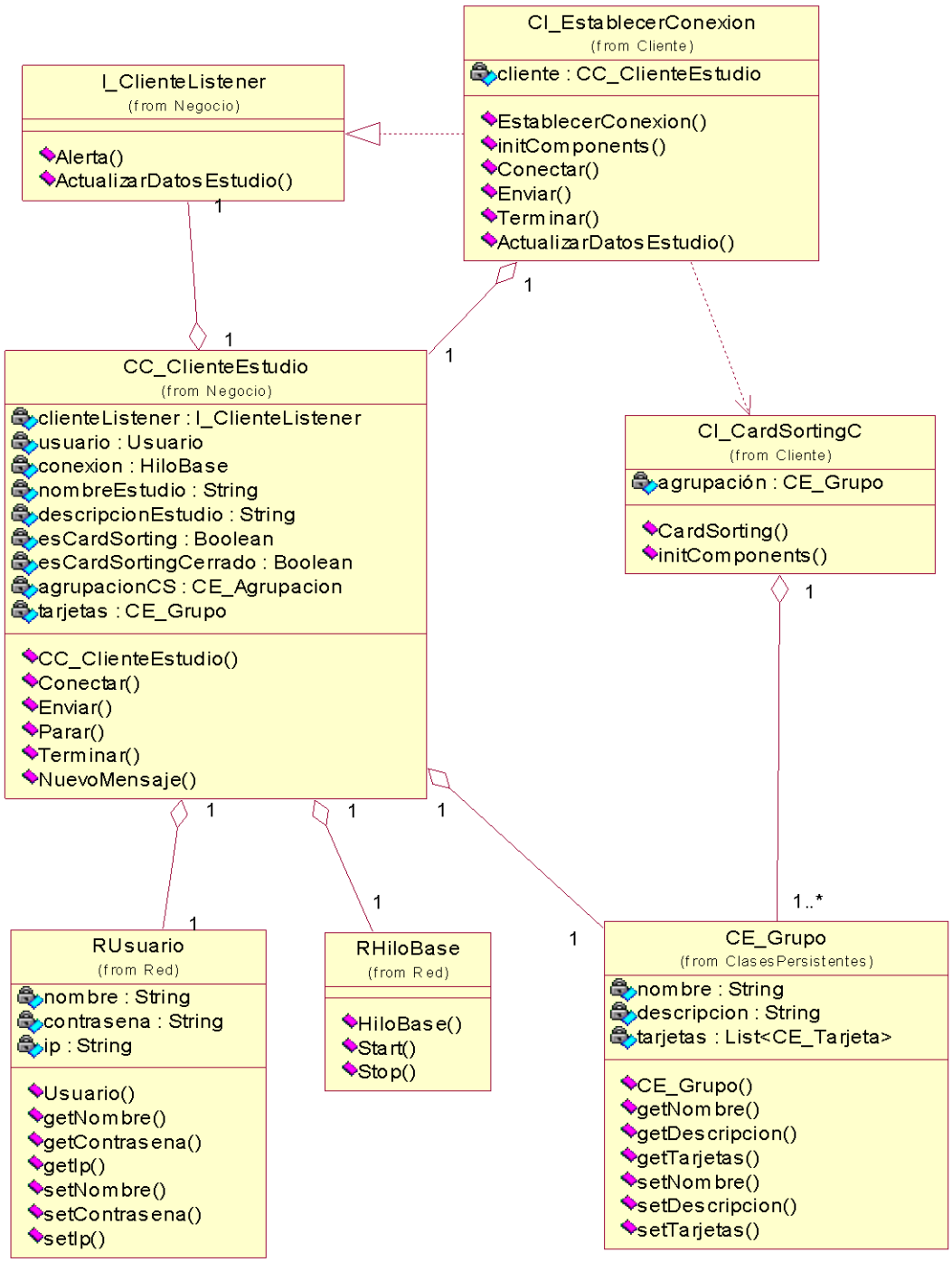


DIAGRAMA DE CLASES DEL DISEÑO – Cu_Realizar_Ordenamientos

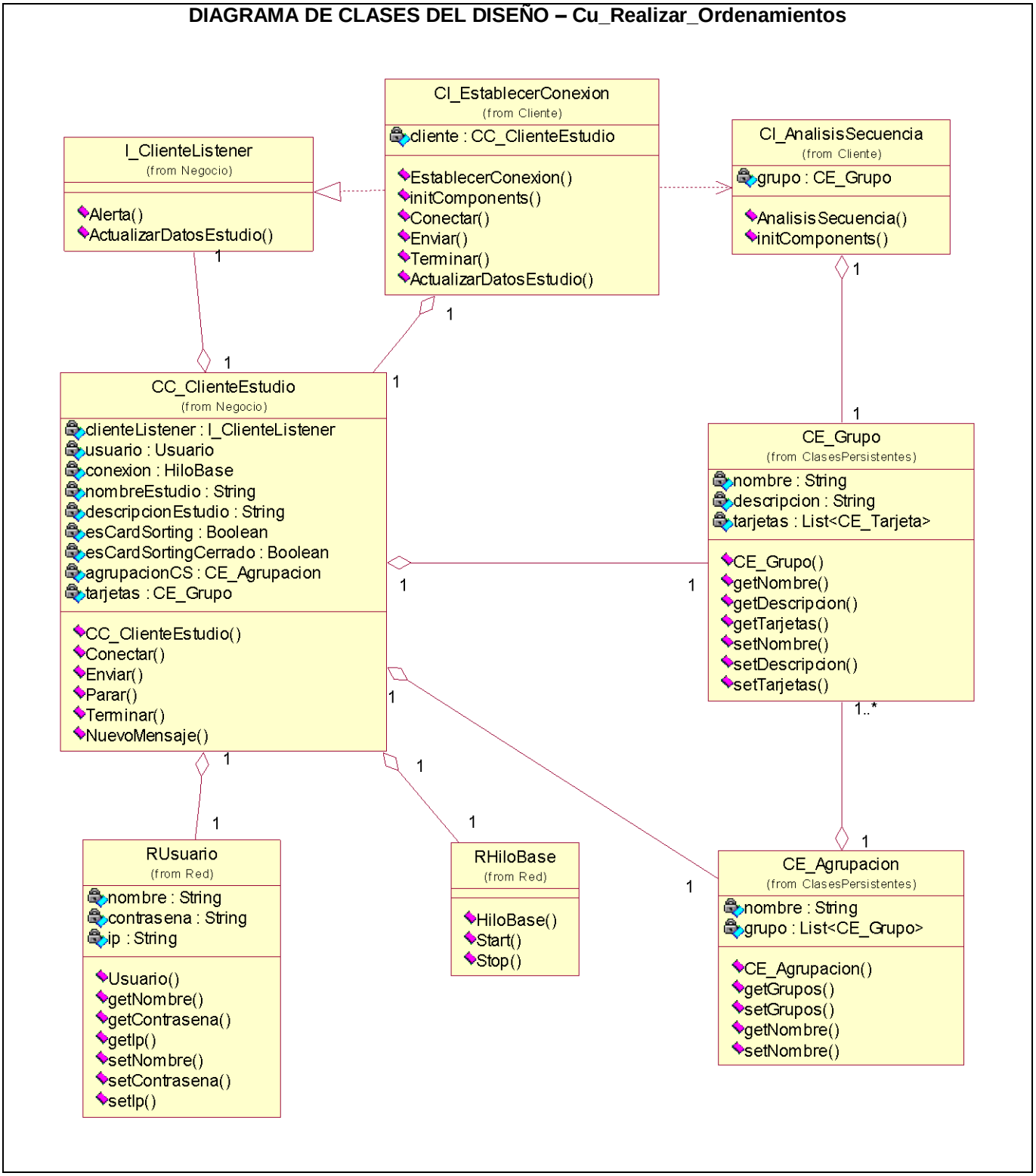
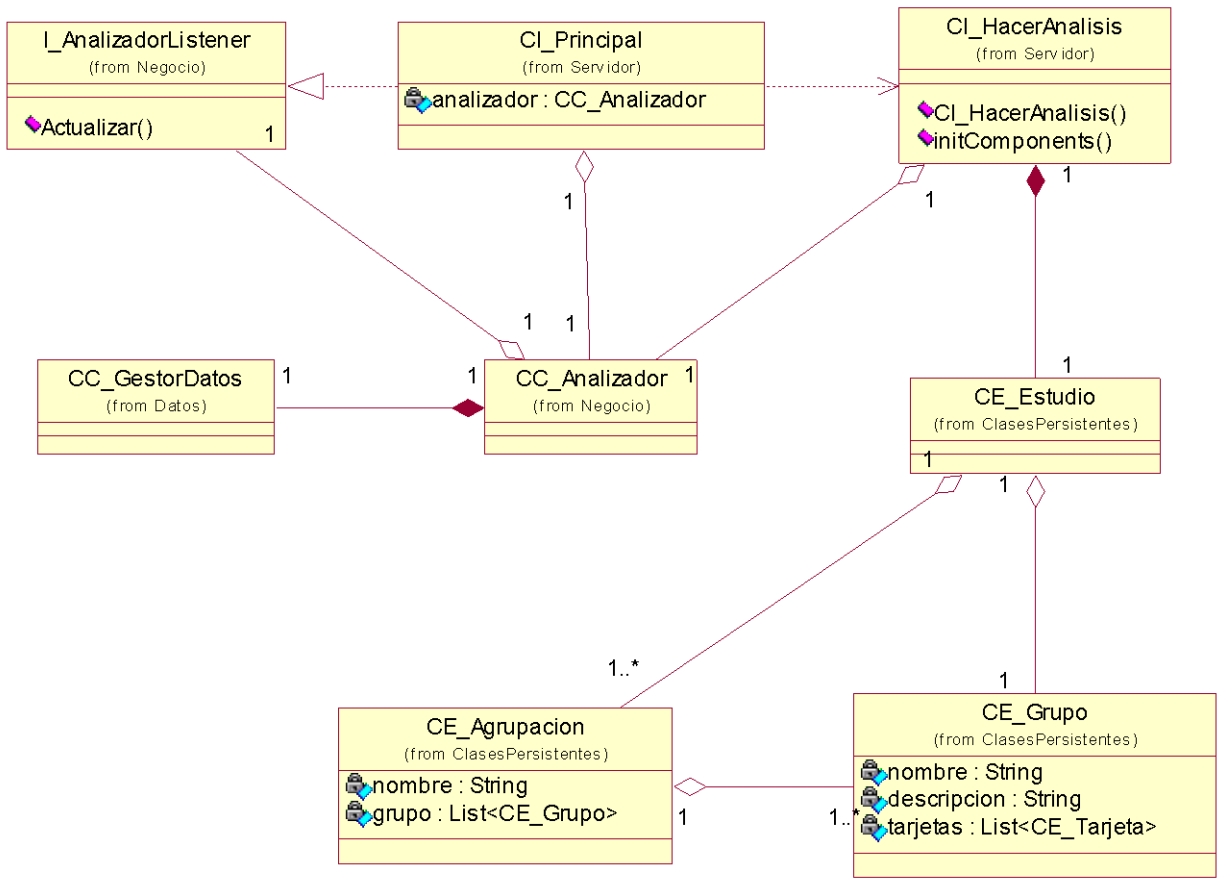


DIAGRAMA DE CLASES DEL DISEÑO – CU_RealizarA_Coocurrencias



3.3.3 Descripción de las clases del diseño

Nombre:	Cl_Principal.java
Descripción:	Vista que contiene el menú principal de la aplicación servidora y muestra la listas de tarjetas existentes y los estudios creados, así como el menú con diferentes opciones.
Nombre:	Cl_GestionarGrupoTarjetas.java
Descripción:	Vista que contiene un formulario mediante el cual el arquitecto crea los diferentes grupos de tarjetas.
Nombre:	Cl_GestionarEstudio.java
Descripción:	Vista que contiene un formulario mediante el cual el arquitecto crea los estudios que

	serán publicados, así como establece cuales serán los grupos en los que se agruparán las tarjetas.
Nombre:	CI_PublicarEstudio.java
Descripción:	Vista que contiene un formulario con los datos requeridos para publicar el estudio y obtener los datos de las conexiones entrantes.
Nombre:	CI_EstablecerConexion.java
Descripción:	Vista de la aplicación cliente que contiene campos para detallar los datos de conexión y establecer la conexión con la aplicación servidora.
Nombre:	CI_CardSorting.java
Descripción:	Vista que contiene el grupo de tarjetas del estudio al que se tuvo acceso y contiene las funcionalidades para realizar las agrupaciones de tarjetas.
Nombre:	CI_AnalisisSecuencia.java
Descripción:	Vista que contiene las agrupaciones creadas y brinda las funcionalidades de ordenarlas.
Nombre:	CI_HacerAnalisis
Descripción:	Vista que contiene los paneles para obtener los resultados de los estudios realizados.

Nombre: CC_Analizador	
Tipo de clase: Controladora	
Atributo	**Tipo**
analizadorListener	I_AnalizadorListener
gestorDatos	CC_GestorDatos
gruposTarjetas	List<String>
estudios	List<String>
Responsabilidades	
Nombre:	CC_Analizador()
Descripción:	Método constructor de la clase.
Nombre:	AdicionarGrupo(CE_Grupo grupo)
Descripción:	Método que adiciona un grupo a la lista de grupos de tarjetas.
Nombre:	ModificarGrupo(CE_Grupo grupoModificado, int pos)

Descripción:	Método que modifica un grupo seleccionado en la lista de grupos de tarjetas.
Nombre:	AdicionarEstudio(CE_Estudio estudio)
Descripción:	Método que adiciona un estudio a la lista de estudios creada y comprueba que dicho estudio no exista.
Nombre:	ModificarEstudio(CE_Estudio estudioModificado, int pos)
Descripción:	Método que modifica un seleccionado de la lista de estudios publicados.
Nombre:	EliminarEstudio(int pos)
Descripción:	Método que elimina un estudio en la lista de estudios creado y la elimina del área de trabajo también.
Nombre:	EliminarGrupoTarjetas(int pos)
Descripción:	Método que elimina un grupo de tarjetas en la lista de tarjetas creada y lo elimina del área de trabajo también.
Nombre:	GuardarGrupo(CE_Grupo grupo)
Descripción:	Método que crea un nuevo grupo, en caso de existir, lo sobre escribe.
Nombre:	GuardarEstudio(CE_Estudio estudio)
Descripción:	Método que crea un nuevo estudio en la lista, en caso de existir lo sobre escribe.
Nombre:	ObtenerGrupo(int pos)
Descripción:	Método que devuelve los datos de un grupo de tarjetas seleccionado de la lista
Nombre:	ObtenerEstudio(int pos)
Descripción:	Método que devuelve los datos de un estudio seleccionado de la lista de estudios.
Nombre:	HacerCardSortng(CE_Estudio est, List<CE_Agrupacion> agrupaciones)
Descripción:	Método que devuelve las agrupaciones resultantes luego de haber obtenido las coocurrencias resultantes de tarjetas por grupos de todos los usuarios que realizaron el estudio.
Nombre:	HacerAnalisisSecuencia(CE_Estudio est, List<CE_Agrupacion> ordenaciones)
Descripción:	Método que devuelve los ordenamientos resultantes luego de haber obtenido las coocurrencias resultantes de grupos y tarjetas ordenados por todos los usuarios que realizaron el estudio.

Nombre: CC_ClienteEstudio	
Tipo de clase: Controladora	
Atributo	**Tipo**

clienteListener	I_ClienteListener
conexion	HiloServidor
nombreEstudio	String
descripcionEstudio	String
esCardSorting	boolean
esCardSortingCerrado	boolean
agrupacionCS	CE_Agrupacion
tarjetas	CE_Grupo
agrupacionAS	CE_Agrupacion
Para cada responsabilidad:	
Nombre:	CC_ClienteEstudio(I_ClienteListener clienteListener)
Descripción:	Método constructor de la clase.
Nombre:	Conectar(String nombre, String contrasena, String ip, int puerto)
Descripción:	Método que establece la conexión de la aplicación cliente con la aplicación servidora a través del IP y puerto especificado. Envía los datos de usuario, contraseña.
Nombre:	Enviar()
Descripción:	Método que envía los el estudio realizado, si es card Sorting envía agrupaciones y si es análisis de secuencia envía un grupo.
Nombre:	Parar()
Descripción:	Método que detiene la conexión con el IP de la aplicación servidor.
Nombre:	Terminar()
Descripción:	Método que se encarga de cerrar la interfaz de establece conexión de la aplicación cliente.
Nombre:	NuevoMensaje(HiloServidor sender, Mensaje mensaje)
Descripción:	Método que interpreta los tipos de mensajes de CONEXIÓN, DESCONEXÓN, PROBLEMAS DE CONEXIÓN, CARD SORTING CERRADO, AGRUPACIÓN o GRUPO DE TARJETAS.
Nombre:	
Descripción:	

Nombre: CC_ServidorEstudio

Tipo de clase: Controladora	
Atributo	**Tipo**
estudio	Estudio
puerto	Int
contraseña	String
hacerCardSorting	boolean
servidor	Servidor
usuariosConectados	List<Usuario>
analizador	CC_Analizador
analizadorListener	I_AnalizadorListener
Responsabilidades:	
Nombre:	CC_ServidorEstudio(CE_Estudio estudio, int puertoPorDefecto)
Descripción:	Método constructor de la clase.
Nombre:	getUsuariosConectados ()
Descripción:	Método que devuelve los usuarios conectados a la aplicación servidora.
Nombre:	ComenzarPublicacion(Int puesto, String contraseña, boolean hacerCardSorting)
Descripción:	Método que publica el estudio en la PC servidora para permitir comenzar la conexión con la PC cliente.
Nombre:	NuevoMensaje(HiloServidor sender, Mensaje mensaje)
Descripción:	Método intérprete de mensajes.

Nombre: Usuario	
Tipo de clase: Controladora	
Atributo	**Tipo**
nombre	String
contraseña	String
ip	String
Para cada responsabilidad:	
Nombre:	Usuario(String nombre, String contrasena, String ip)
Descripción:	Método constructor de la clase.
Nombre:	getContrasena ()

Descripción:	Método que devuelve la contraseña del usuario.
Nombre:	setContrasena(String contrasena)
Descripción:	Método que establece la contraseña del usuario
Nombre:	getIp()
Descripción:	Método que nos devuelve el Ip del que se conecta el usuario
Nombre:	setIp(String ip)
Descripción:	Método que establece el Ip del usuario
Nombre:	getNombre()
Descripción:	Método que devuelve el nombre del usuario.
Nombre:	setNombre(String nombre)
Descripción:	Método que establece el nombre del usuario

Nombre: Servidor extends HiloBase	
Tipo de clase: Controladora de Red	
Atributo	**Tipo**
interprete	IManipuladorMensajes
Contraseña	String
port	Int
cantMaxConexiones	Int
cantMaxEnEspera	Int
cantEnEspera	Int
Conexiones	List<HiloServidor>
Para cada responsabilidad:	
Nombre:	Servidor(IManipuladorMensajes interprete, int port, int cantMaxConexiones, String contrasena)
Descripción:	Constructor de la clase
Nombre:	run()
Descripción:	Método donde se ejecutan los hilos.
Nombre:	NuevoMensaje(HiloServidor sender, Mensaje mensaje)
Descripción:	Método que implementa a la clase IManipuladorMensajes.

Nombre: HiloServidor extends HiloBase	
Tipo de clase: Controladora de Red	
Atributo	**Tipo**
usuario	Usuario
socket	Socket
interprete	IManipuladorMensajes
cantMaxConexiones	Int
br	BufferedReader
bw	BufferedWriter
Para cada responsabilidad:	
Nombre:	HiloServidor(Socket socket, IManipuladorMensajes interprete)
Descripción:	Operación que devuelve una instancia de la clase con socket y un intérprete. Usado para cuando se recibe un socket en el servidor.
Nombre:	HiloServidor(Usuario usuario, String ipRemoto, int puertoRemoto, IManipuladorMensajes interprete)
Descripción:	Operación que devuelve una instancia de la clase con un usuario, un puerto y un intérprete. Usado para cuando se inicializa en el cliente.
Nombre:	setUsuario(Usuario usuario)
Descripción:	Método para introducir los valores de un usuario.
Nombre:	getUsuario()
Descripción:	Método que devuelve el usuario.
Nombre:	MandarMensaje(Mensaje mensaje)
Descripción:	Envía los mensajes a los sockets en espera.

Nombre: Mensaje	
Tipo de clase: Controladora de Red	
Atributo	**Tipo**
tipo	TiposMensaje
valor	Object
Para cada responsabilidad:	
Nombre:	Mensaje(TiposMensaje tipo, Object valor)

Descripción:	Constructor que devuelve una instancia de la clase con el valor de un mensaje.
Nombre:	getTipo()
Descripción:	Método que devuelve el tipo de mensaje específico.
Nombre:	setTipo(TiposMensaje tipo)
Descripción:	Método que permite introducir el tipo de mensaje.
Nombre:	getValor()
Descripción:	Método que devuelve el valor del mensaje.
Nombre:	setValor(Object valor)
Descripción:	Método para entra los valores

Nombre: HiloBase	
Tipo de clase: Controladora de Red	
Atributo	**Tipo**
Hilo	Thread
Para cada responsabilidad:	
Nombre:	HiloBase()
Descripción:	Constructor de la clase.
Nombre:	Start()
Descripción:	Es el método que es llamado para poner un hilo en escucha.
Nombre:	Stop()
Descripción:	Es el método llamado para detener el hilo escucha.

Nombre: IManipuladorMensajes	
Tipo de clase: Interface	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**NuevoMensaje(HiloServidor sender,Mensaje mensaje)**
Descripción:	Manipula los tipos de mensajes que se envían o se reciben.

Nombre: I_ClienteListener
Tipo de clase: Interface

Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Alerta(String texto)
Descripción:	Método que se actualiza cuando hay algún problema de conexión.
Nombre:	ActualizarDatosEstudio(String nombreEstudio, String DescripcionEstudio)
Descripción:	Método que actualiza el nombre y la descripción de un estudio cuando el cliente se conecta a el.

Nombre: CC_GestorDatos	
Tipo de clase: controladora	
Atributo	**Tipo**
areaTrabajo	String
codificación	String
file	File
Para cada responsabilidad:	
Nombre:	**public CC_GestorDatos(String areaTrabajo, String codificacion)**
Descripción:	Constructor de la clase.
Nombre:	ListarGruposTarjetas()
Descripción:	Método encargado de listar los grupos de tarjetas creados.
Nombre:	ListarEstudios()
Descripción:	Método que se encarga de listar los estudios creados por el arquitecto.
Nombre:	GuardarGrupoTarjetas(CE_Grupo grupo)
Descripción:	Método que guarda en grupos todas las tarjetas que va creado el arquitecto.
Nombre:	GuardarEstudio(CE_Estudio estudio)
Descripción:	Método que manda a guardar los estudios creados por el arquitecto.
Nombre:	ObtenerGrupoTarjetas(String nombre)
Descripción:	Método que devuelve las tarjetas de un determinado grupo de tarjetas seleccionado de la lista.
Nombre:	ObtenerEstudio(String nombre)
Descripción:	Método que permite obtener los datos de un determinado estudio seleccionado de la lista de estudios creados.

Nombre:	EliminarGrupoTarjetas(String nombre)
Descripción:	Método que partiendo del grupo de tarjetas seleccionado de la lista permite que este sea eliminado.
Nombre:	EliminarEstudio(String nombre)
Descripción:	Método que partiendo del estudio que sea seleccionado de la lista de estudios creada, permite que este sea eliminado,

Nombre: CE_Tarjeta	
Tipo de clase: entidad	
Atributo	**Tipo**
nombre	String
descripcion	String
Responsabilidades	
Nombre:	public CE_Tarjeta(String nombre, String descripcion)
Descripción:	Operación que devuelve una instancia de la clase con una dirección IP asociada a la misma.
Nombre:	getNombre ()
Descripción:	Método para obtener el nombre de una tarjeta.
Nombre:	setNombre(String nombre)
Descripción:	Método para entra los valores del nombre de una tarjeta
Nombre:	getDescripcion()
Descripción:	Método para obtener la descripción de una tarjeta.
Nombre:	setDescripcion(String descripcion)
Descripción:	Método para entrar los valores de la descripción de una tarjeta.

Nombre: CE_Estudio	
Tipo de clase: entidad	
Atributo	**Tipo**
nombre	String
descripcion	String

contrasena	String
cerrado	boolean
tarjetas	CE_Grupo
plantilla	CE_Agrupacion
agrupaciones	List<CE_Agrupacion>
resultadoCardSorting	CE_Agrupacion
resultadoCSOrdenado	CE_Agrupacion
ordenamientos	List<CE_Agrupacion>

Responsabilidades	
Nombre:	CE_Estudio(String nombre, String contrasena, String descripción, boolean cerrado)
Descripción:	Operación que devuelve una instancia de la clase con un nombre, una contraseña específica para el estudio y una descripción.
Nombre:	CE_Estudio(String nombre, String contrasena, String descripcion, boolean cerrado, CE_Grupo tarjetas)
Descripción:	Operación que devuelve una instancia de la clase incluyendo las anteriores características más un grupo de tarjetas asociado a la misma.
Nombre:	getAgrupaciones()
Descripción:	Operación que devuelve las agrupaciones de un estudio.
Nombre:	isCerrado()
Descripción:	Operación que verdadero cuando se desea realizar Card Sorting de tipo cerrado en el estudio.
Nombre:	setCerrado(boolean cerrado)
Descripción:	Método para entrar la especificación de cuando el card Sorting es de tipo cerrado.
Nombre:	getNombre()
Descripción:	Método que devuelve el nombre del estudio.
Nombre:	setNombre(String nombre)
Descripción:	Método para darle un valor al nombre del estudio.
Nombre:	getDescripcion()
Descripción:	Método que devuelve la descripción de un estudio.
Nombre:	setDescripcion(String descripcion)
Descripción:	Método para entrar valores a la descripción de un estudio.
Nombre:	getContrasena()

Descripción:	Método que permite obtener la contraseña de un estudio determinado.
Nombre:	setContrasena(String contrasena)
Descripción:	Método para entrar los valores de la contraseña de un estudio.
Nombre:	getPlantilla()
Descripción:	Método que permite obtener los valores de una plantilla de grupos creado cuando se trata de realizar card Sorting de tipo cerrado
Nombre:	setPlantilla(CE_Agrupacion plantilla)
Descripción:	Método que permite introducir los valores de las plantillas. Consiste en la creación de los grupos iniciales cuando se trata de card Sorting cerrado.
Nombre:	getResultadoCSOrdenado()
Descripción:	Método que permite obtener los resultados de card Sorting luego de haber realizado las agrupaciones en los grupos iniciales creados por defectos.
Nombre:	setResultadoCSOrdenado(CE_Agrupacion resultadoCSOrdenado)
Descripción:	Método que permite introducir un valor de tipo agrupación del card Sorting ordenado.
Nombre:	getResultadoCardSorting()
Descripción:	Método que permite obtener los resultados de card Sorting una vez realizada las agrupaciones por los usuarios.
Nombre:	setResultadoCardSorting(CE_Agrupacion resultadoCardSorting)
Descripción:	Método que permite introducir un valor de tipo agrupación del card Sorting.
Nombre:	getTarjetas()
Descripción:	Método para obtener una tarjeta de un grupo de tarjetas específico.
Nombre:	setTarjetas(CE_Grupo tarjetas)
Descripción:	Método para entrar los valores de una tarjeta de un grupo determinado.

Nombre: CE_Grupo	
Tipo de clase: entidad	
Atributo	**Tipo**
nombre	String
descripcion	String
tarjetas	List<CE_Tarjeta>

Responsabilidades	
Nombre:	CE_Grupo(String nombre, String descripcion)
Descripción:	Operación que devuelve una instancia de la clase con un nombre y una descripción.
Nombre:	getNombre ()
Descripción:	Método para obtener el nombre de un grupo de tarjetas.
Nombre:	setNombre(String nombre)
Descripción:	Método para entrar el valor de un nombre del grupo
Nombre:	getDescripcion()
Descripción:	Método que devuelve la descripción de un grupo.
Nombre:	setDescripcion(String descripcion)
Descripción:	Método para entrar valores a la descripción de un grupo.
Nombre:	getTarjetas()
Descripción:	Método que permite obtener las tarjetas de un grupo.
Nombre:	setTarjetas(List<CE_Tarjeta> tarjetas)
Descripción:	Método que permite darle valores a todas las tarjetas de un grupo.
Nombre:	getTarjetas()
Descripción:	Método para obtener una tarjeta de un grupo de tarjetas específico.

Nombre: CE_Agrupacion	
Tipo de clase: entidad	
Atributo	**Tipo**
nombre	String
Grupos	List<CE_Grupo>
Responsabilidades	
Nombre:	CE_Agrupacion(String nombre)
Descripción:	Operación que devuelve una instancia de la clase con un nombre de la agrupación.
Nombre:	getNombre()
Descripción:	Método para obtener el nombre de una agrupación.
Nombre:	setNombre(String nombre)
Descripción:	Método para entrar el valor de una agrupación.
Nombre:	getGrupos()

Descripción:	Método que devuelve las grupos de una agrupación.
Nombre:	setGrupos(List<CE_Grupo> grupos)
Descripción:	Método para entrar los valores de los grupos de una agrupación.

3.4 Conclusiones.

En este capítulo se ubica al lector en cómo el análisis muestra una visión detallada del sistema. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales. Se definieron los diagramas de clases del análisis y diseño para aquellos procesos que responden a funcionalidades de gran importancia para el sistema, para cada uno de estos se elaboraron los diagramas de secuencia para representar el comportamiento entre las clases del diseño, así como la descripción de cada una de estas.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

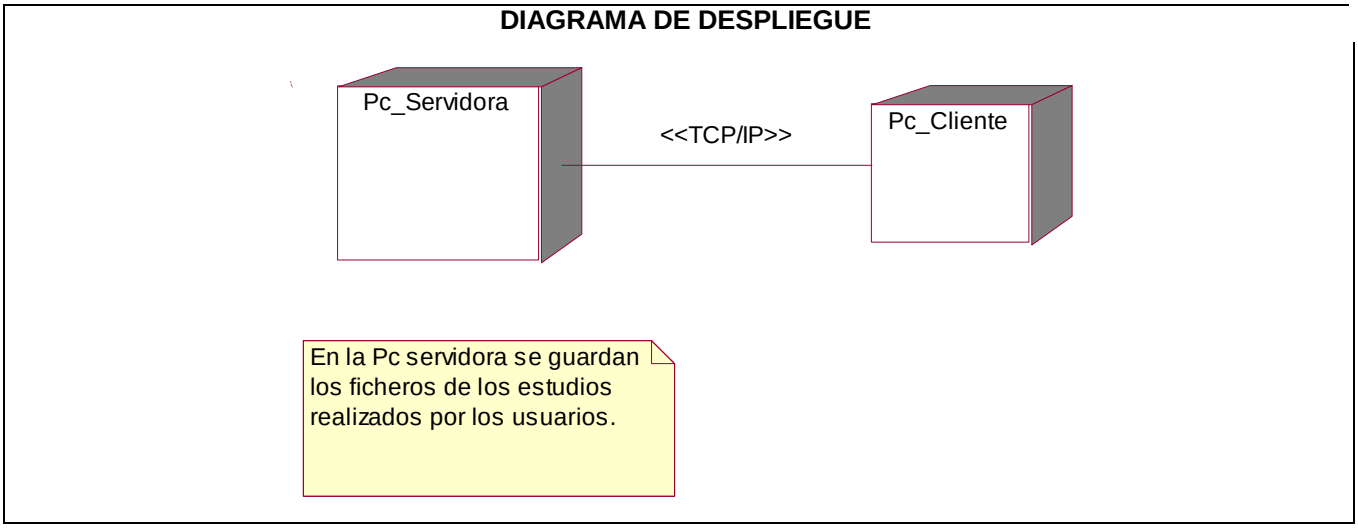
4.1 Introducción

En este capítulo se presentan los flujos de trabajo correspondientes a la implementación y pruebas. Se comenzará con implementación partiendo de los resultados obtenidos en el diseño, para dar entrada a la etapa de pruebas, en esta última se realizará el diseño de los casos de pruebas para los casos de de usos que se determinaron de gran importancia para el sistema en general.

4.2 Implementación

El flujo de trabajo de implementación detalla cómo los elementos del modelo del diseño se implementan en términos de componentes y representa cómo se organizan en el modelo de despliegue. Permite obtener los diagramas de despliegue y componentes los cuales son artefactos que conforman lo que se conoce como un modelo de implementación. A continuación se realizará el diagrama de despliegue y diagrama de componentes correspondiente a la lógica de presentación y negocio del software informático para uso de los arquitectos de información en la UCI.

4.2.1 Diagrama de despliegue



4.2.2 Diagrama de componentes

DIAGRAMA DE COMPONENTES-LOGICA DE PRESENTACION

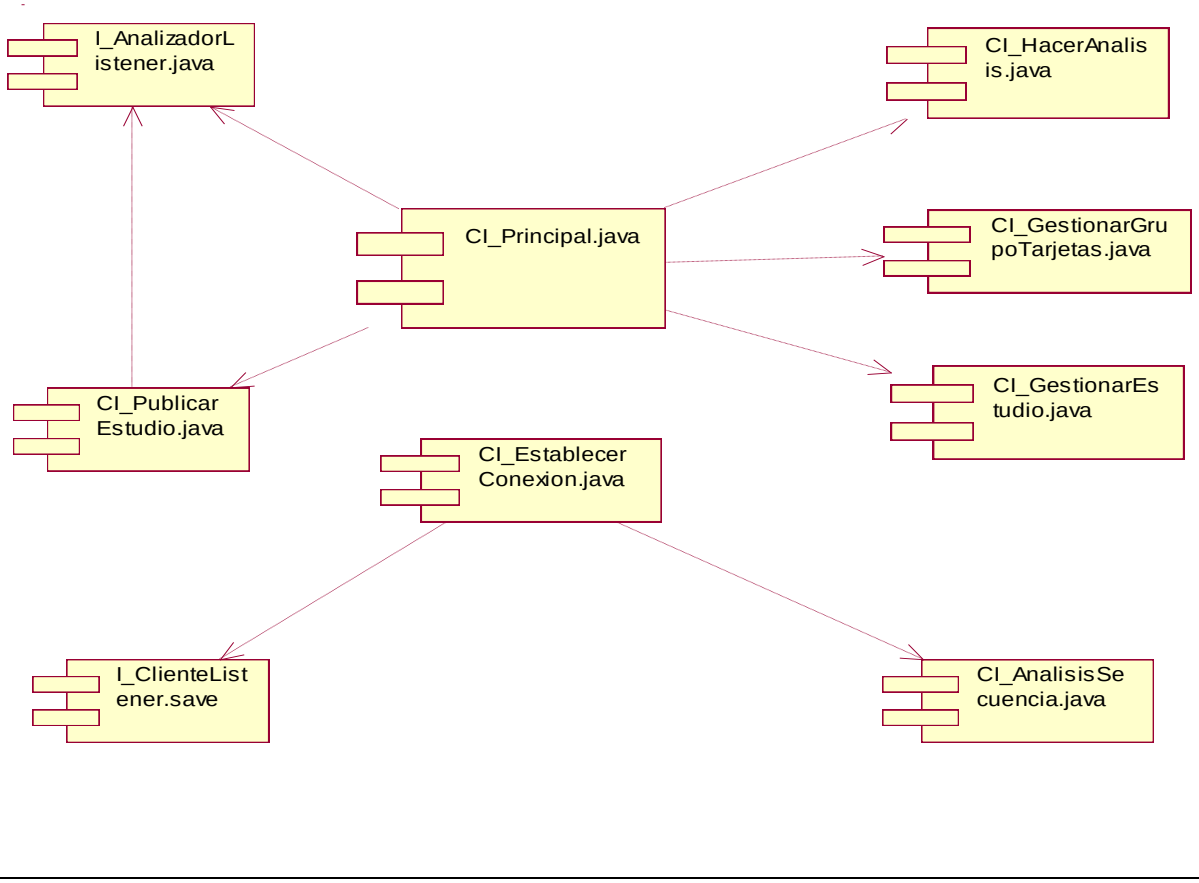


DIAGRAMA DE COMPONENTES-LOGICA DE NEGOCIO

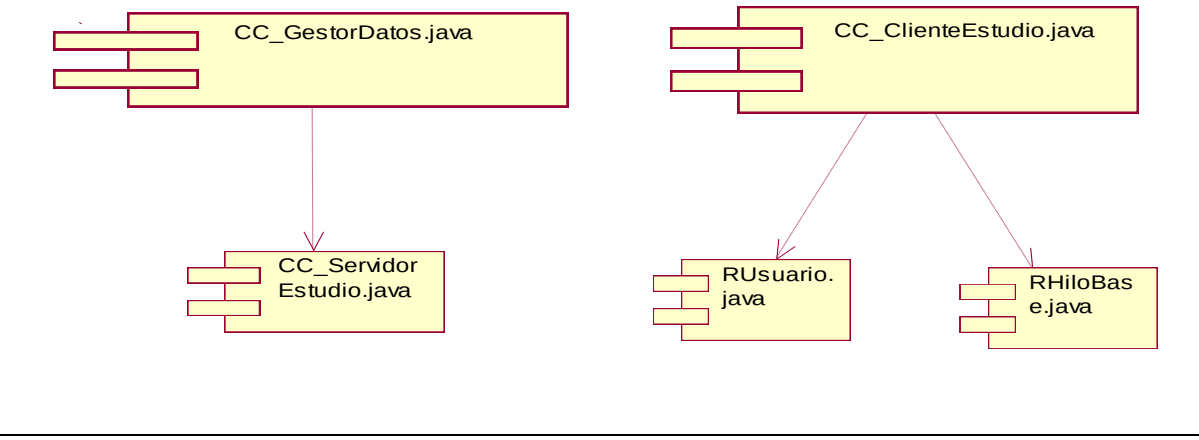


DIAGRAMA DE COMPONENTES-LOGICA DE DATOS

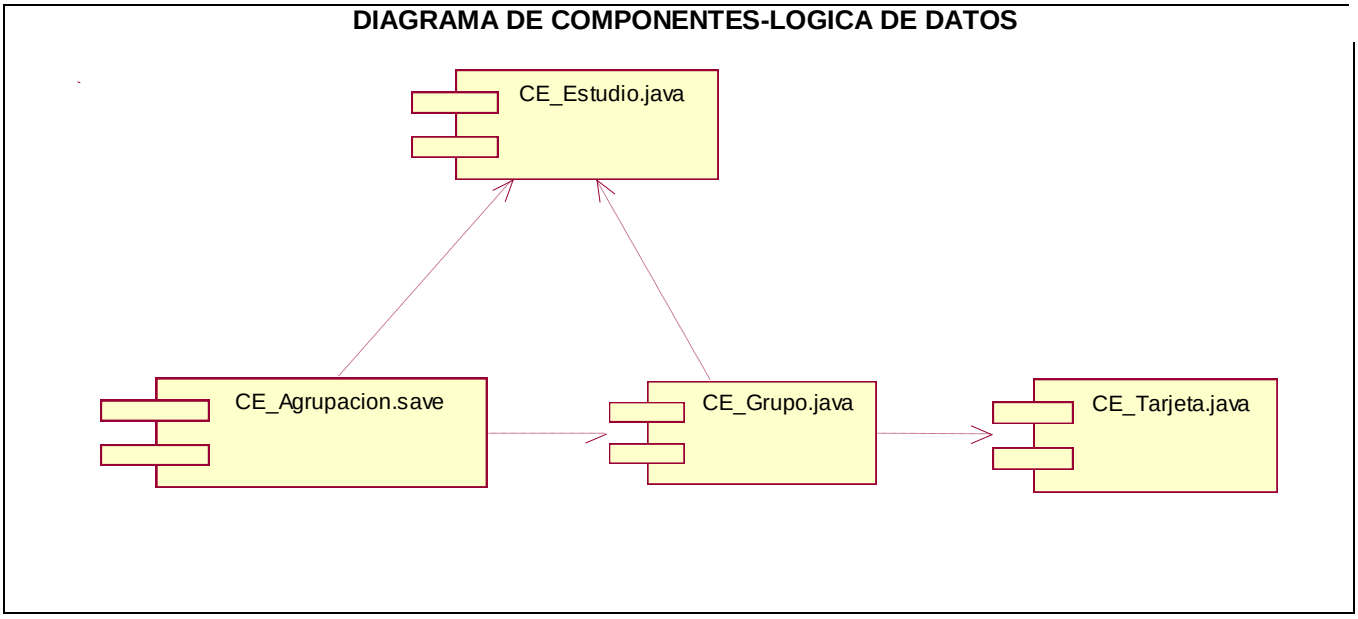
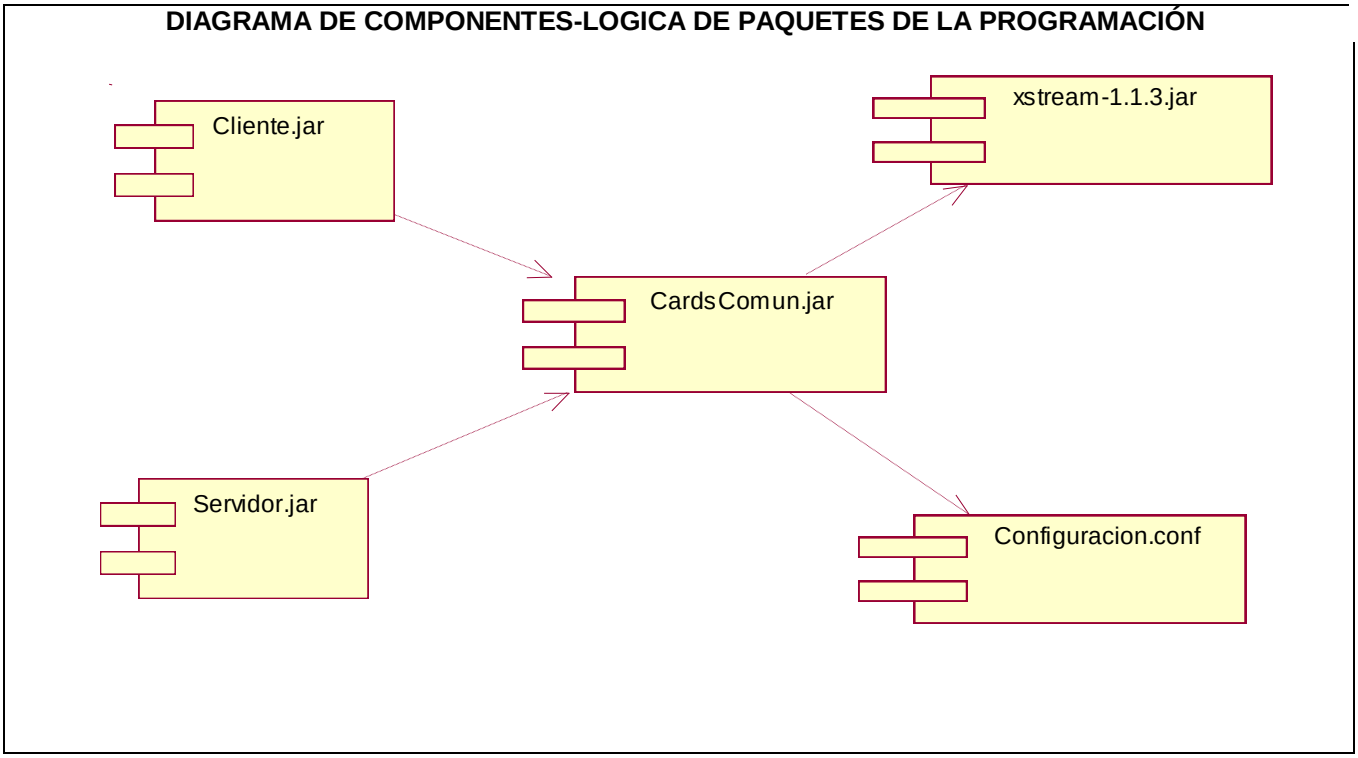


DIAGRAMA DE COMPONENTES-LOGICA DE PAQUETES DE LA PROGRAMACIÓN



4.3 Modelo de Prueba.

Todo desarrollo de software ha de ir acompañado de una actividad que garantice la calidad del mismo, por lo que la prueba de un software es un elemento crítico para su garantía y calidad. Representa una revisión final de las especificaciones del diseño y de la codificación. Se definirán a continuación los diseños de los casos de pruebas para los casos de usos significativos para la arquitectura del software.

4.3.1 Diseño de los casos de usos de prueba

Nombre del caso de uso: Gestionar_tarjetas. (Crear y modificar grupo de tarjeta)

Entrada	**Resultados**	**Condiciones**
El Arquitecto introduce un nombre del grupo de tarjetas	El sistema muestra un mensaje: "¿Confirma que desea guardar El grupo?"	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de crear grupo de tarjetas.
El Arquitecto deja el campo del nombre del grupo en blanco.	El Sistema muestra el mensaje "Debe entrar un nombre para el grupo de tarjetas". El Sistema muestra el formulario para insertar el nombre del grupo.	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de crear grupo de tarjetas.
El Arquitecto introduce el grupo: "Grupo de tarjetas No 1"	El Sistema muestra el mensaje "El grupo de tarjetas ya existe". El Sistema muestra el formulario para insertar el nombre del grupo.	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de crear grupo de tarjetas.
El Arquitecto introduce el grupo: "Grupo de tarjetas No 1"	El Sistema muestra el mensaje "El grupo de tarjetas ya existe". ¿Desea sobre escribir el grupo existente por el nuevo?	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de crear grupo de tarjetas.
El arquitecto introduce la tarjeta:	El Sistema muestra el mensaje "Tarjeta existente, ¿Desea sobre escribir la	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación

"Tarjeta No1"	descripción de la tarjeta?".	servidora la opción de crear grupo de tarjetas.

Nombre del caso de uso: Gestionar_Tarjetas. (Eliminar Grupo de tarjetas)

Entrada	**Resultados**	**Condiciones**
El Arquitecto selecciona de la lista de grupos de tarjetas un grupo a eliminarse	El sistema solícita confirmación de eliminar el grupo. "¿Confirma que desea eliminar el grupo seleccionado?"	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de eliminar grupo de tarjetas.
El arquitecto ejecuta la acción de eliminar un grupo de tarjetas.	El Sistema muestra un mensaje. " UD. Debe seleccionar un grupo a eliminar". El Sistema muestra la lista de grupos de tarjetas	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de eliminar grupo de tarjetas.

Nombre del caso de uso: Gestionar Estudios. (Crear y modificar estudio)

Entrada	**Resultados**	**Condiciones**
El Arquitecto introduce un nombre del estudio.	El sistema muestra un mensaje: "¿Confirma que desea guardar El estudio?"	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de crear estudio.
El Arquitecto deja el campo del nombre del grupo en blanco.	El Sistema muestra el mensaje "Debe entrar un nombre para el estudio". El Sistema muestra el formulario para insertar el nombre del estudio.	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de crear estudio.
El Arquitecto introduce el estudio: "Estudio No 1"	El Sistema muestra el mensaje "El estudio ya existe". El Sistema muestra el formulario para insertar el nombre del estudio.	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de crear estudio.
El Arquitecto introduce el	El Sistema muestra el mensaje "El estudio de tarjetas ya existe". ¿Desea	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de crear estudio.

estudio: “Estudio No 1”	sobre escribir el estudio existente por el nuevo?	
El arquitecto introduce el grupo: “Grupo 1”	El Sistema muestra el mensaje “Grupo existente, ¿Desea sobre escribir la descripción del grupo?”.	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de crear estudio.

Nombre del caso de uso: Gestionar Estudios. (Eliminar estudio)

Entrada	**Resultados**	**Condiciones**
El Arquitecto selecciona de la lista de estudios creada, un estudio eliminarse	El sistema solícita confirmación de eliminar el estudio. “¿Confirma que desea eliminar el estudio seleccionado?”	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de eliminar estudio.
El Arquitecto selecciona de la lista de estudios creada, un estudio eliminarse	El Sistema muestra un mensaje. ” UD. Debe seleccionar un estudio a eliminar”. El Sistema muestra la lista de estudios.	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de eliminar estudio.

Nombre del caso de uso: Publicar estudio.

Entrada	**Resultados**	**Condiciones**
El Arquitecto deja en blanco el campo para especificar el puerto.	El sistema muestra el mensaje “¿Debe especificar el puerto de conexión?”	El Arquitecto debe haber seleccionado en la interfaz Publicar estudio la opción de “Comenzar”.
El Arquitecto deja en blanco el campo para especificar la contraseña.	El sistema muestra el mensaje “¿Debe especificar la contraseña de la publicación?”	El Arquitecto debe haber seleccionado en la interfaz Publicar estudio la opción de “Comenzar”.

Nombre del caso de uso: Establecer_Conexión.

Entrada	**Resultados**	**Condiciones**
El usuario introduce nombre, Ip y contraseña de la PC servidora	El sistema muestra un mensaje: "¿Desea establecer la conexión?"	El Usuario debe presionar el botón conectar.
El usuario deja campos en blanco.	El Sistema muestra el mensaje "Error en la entrada de datos". El Sistema muestra el formulario para insertar los datos de conexión.	El Usuario debe presionar el botón conectar.
El usuario introduce mal la contraseña	El Sistema muestra el mensaje "contraseña errónea intente de nuevo". El Sistema muestra un formulario para insertar los datos de la contraseña.	El Usuario debe presionar el botón conectar.
El usuario introduce 163.875.34.343 En el campo de IP	El Sistema muestra el mensaje "IP incorrecto, introduzca un IP válido". El Sistema muestra el formulario para insertar los datos del la dirección IP.	El Usuario debe presionar el botón conectar.
El Usuario pone un nombre de usuario ya conectado	El sistema muestra un mensaje: "Ya existe un usuario con ese nombre conectado" El sistema muestra el formulario para introducir un nuevo nombre de usuario.	El Usuario debe presionar el botón conectar.

Nombre del caso de uso: Realizar_Agrupaciones.

Entrada	**Resultados**	**Condiciones**
El usuario selecciona una tarjeta pero no selecciona ningún un grupo de la	El sistema muestra un mensaje: "Ud. debe seleccionar un grupo para adicionar la tarjeta seleccionada"	El usuario debe presionar el botón "<<" (Adicionar)

lista.		
El usuario selecciona un grupo de la lista pero no selecciona ninguna tarjeta.	El sistema muestra un mensaje: "Ud. debe seleccionar una tarjeta para adicionarla en el grupo seleccionado"	El usuario debe presionar el botón "<<" (Adicionar)
No hay tarjetas en la lista.	El sistema muestra un mensaje: "No existen mas tarjetas en la lista. Ud. a terminado de repartir todas las tarjetas"	El usuario debe presionar el botón "<<" (Adicionar)
El usuario selecciona un grupo pero no selecciona ninguna tarjeta de ese grupo.	El sistema muestra un mensaje: "Ud. debe seleccionar una tarjeta eliminarla del grupo"	El usuario debe presionar el botón ">>" (Eliminar)
No hay tarjetas en el grupo seleccionado.	El sistema muestra un mensaje: "No existen tarjetas en el grupo seleccionado"	El usuario debe presionar el botón ">>" (Eliminar
El usuario no selecciona ningún grupo.	El sistema muestra un mensaje: "Ud. debe seleccionar un grupo de tarjetas"	El usuario debe presionar el botón ">>" (Eliminar)

Nombre del caso de uso: Realizar_Ordenamientos.

Entrada	**Resultados**	**Condiciones**
El usuario NO selecciona un grupo de la lista.	El sistema muestra un mensaje: "UD debe seleccionar un grupo a ordenar"	El usuario debe presionar el botón "Subir" (para ordenar los grupos)
El usuario NO selecciona un grupo de la lista.	El sistema muestra un mensaje: "UD debe seleccionar un grupo a ordenar"	El usuario debe presionar el botón "Bajar" (para ordenar los grupos)
El usuario NO	El sistema muestra un mensaje: "UD	El usuario debe presionar el botón "Subir"

selecciona una tarjeta del grupo seleccionado.	debe seleccionar una tarjeta del grupo para poder ordenarla"	(para ordenar las tarjetas)
El usuario NO selecciona una tarjeta del grupo seleccionado.	El sistema muestra un mensaje: "UD debe seleccionar una tarjeta del grupo para poder ordenarla"	El usuario debe presionar el botón "Bajar" (para ordenar las tarjetas)

Nombre del caso de uso: RealizarA_coocurrencias.

Entrada	**Resultados**	**Condiciones**
El Arquitecto presiona el botón de "blanquear" o eliminar estudio realizado.	El sistema muestra un mensaje: "¿Confirma que desea eliminar los resultados del estudio seleccionado?"	El Arquitecto debe haber seleccionado en la interfaz principal del la aplicación servidora la opción de obtener resultados del estudio.

4.4 Conclusiones

En este capítulo se obtuvieron como resultado los diagramas de componentes así como el diagrama de despliegue y el diseño de los casos de usos de prueba de los procesos más importantes respectivos a la arquitectura del software realizado.

CONCLUSIONES

El proceso de realización del presente trabajo ha atravesado diferentes etapas, cumpliéndose todos los objetivos del mismo. En la etapa inicial se investigó sobre muchos de los softwares que implementan técnicas de co-ocurrencias, además de las bases tecnológicas para resolver el problema planteado. Después se realizó en una segunda etapa el análisis y diseño en la que se obtuvieron los artefactos necesarios para llegar a la solución deseada, dando paso a la fase de implementación de la aplicación propuesta. Con el objetivo de verificar la calidad y las diferentes funcionalidades que el sistema debe cumplir, se llevó a cabo la realización de las pruebas del sistema.

Al finalizar este trabajo se logró la implementación de una aplicación para los Arquitectos de Información de la Universidad de las Ciencias Informáticas, para su uso y mejoramiento del trabajo de los mismos. Con la puesta práctica de esta aplicación desktop se espera poder facilitar el trabajo de los Arquitectos de Información de la UCI, haciendo su labor más rápida y eficiente, de manera que pueda evidenciarse a través de mejoras en el proceso de creación de sitios web o multimedia, para lograr satisfacer al máximo las necesidades del usuario final del producto.

RECOMENDACIONES

- ✓ Continuar con la investigación para aumentar las funcionalidades del sistema, con el objetivo de obtener mejoras en futuras versiones del mismo.
- ✓ Utilizar diferentes tipos de análisis de cluster para futuras versiones del software.
- ✓ Poner a prueba el Sistema durante un período de tiempo significativo, para comprobar su desempeño.
- ✓ Implementar un sistema web que brinde las funcionalidades existentes y las que pudieran aparecer durante el periodo de prueba piloto de la presente versión realizada como aplicación de escritorio.

REFERENCIA BIBLIOGRÁFICA

1 **Wurman, Richard Saul. 1975.** *Information Architects.* 1975.

2 *Arquitectura de la información: algo más que un concepto.* **Reyes, Gómez. 2002.** La Habana : s.n., 2002. En: Congreso Internacional de Información INFO 2002.

3 **Yusef Hassan Montero, Francisco J. Martín Fernández. 2003.** Qué es la Accesibilidad Web. *nosolousabilidad.com.* [En línea] 14 de julio de 2003. http://www.nosolousabilidad.com/articulos/accesibilidad.htm.

4 **Gómez, n Carlos García. 2004.** *Usabilidad de las páginas de inicio de los diarios digitales españoles.* [En línea] julio-diciembre de 2004. http://ibersid.eu/ojs/index.php/scire/article/viewFile/1499/1477.

5 **Peñalver, Gregorio Hernández. 2007.** DEPARTAMENTO DE MATEMÁTICA APLICADA. *dma.fi.upm.es.* [En línea] 8 de octubre de 2007. http://www.dma.fi.upm.es/gregorio/JavaGC/Cconvexo/teoriaJava.html.

BIBLIOGRAFÍA

1. **JAMES RAMBAUGH, I. J., GRADY BOOCH. 1998.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* [ed.] A Wesley. 1998. pág. http://bibliodoc.uci.cu/pdf/reg03050.pdf.

2. **MARCUCCI, A. GAUTHIER. 2005.** *Implementación y Evaluación de Métedos de Clasificación: Estadísticos, de Agrupamiento, Árboles de Decisión y MetaHeurísticos.* 2005.

3. **PRESSMAN, R. 2002.** *Ingeniería del Software. Un enfoque práctico.* s.l. : Hill, M, 2002. pág. http://bibliodoc.uci.cu/pdf/reg02689.pdf.

4. **SCHMULLER, J. 2000.** *Aprendiendo UML en 24 Horas.* s.l. : Education, P, 2000. pág. http://bibliodoc.uci.cu/pdf/reg00004.pdf.

5. **2004.** *UML y Patrones: Introducción al análisis y diseño orientado a objetos.* La Habana : s.n., 2004. pág. 327.

6. **WENDY BOGGS, M. B. 2002.** *UML with Rational Rose 2002.* [ed.] Sybex. 2002. pág. http://bibliodoc.uci.cu/pdf/0782140173.pdf.

7. **Camus, Juan C. 2004.** ¿Qué es la Arquitectura de Información? *usando.info.* [En línea] 09 de 04 de 2004. http://www.usando.info/main_file.php/us_investiga/8485.

8. **León, Rodrigo Ronda. 2003-2007.** Análisis de Secuencia: una herramienta para la Arquitectura de Información. *nosolousabilidad.* [En línea] 2003-2007. http://www.nosolousabilidad.com/articulos/analisis_secuencia.htm.

9. La diagramación en la arquitectura de información. *nosolousabilidad.* [En línea] http://www.nosolousabilidad.com/articulos/diagramacion.htm.

10. *Desarrollo Conceptual y la técnica de Card Sorting.* **Santamaría, Sergio Ortega. 2005.** 1886-8592, 14 de Diciembre de 2005, No Solo Usabilidad.

11. *Card Sorting: Técnica de categorización de contenidos.* **Montero, Yusef Hassan. 2004.** 1886-8592, Granada : s.n., 23 de marzo de 2004, No Solo Usabilidad.

12. **Wurman, Richard Saul. 1975.** *Information Architects.* 1975.

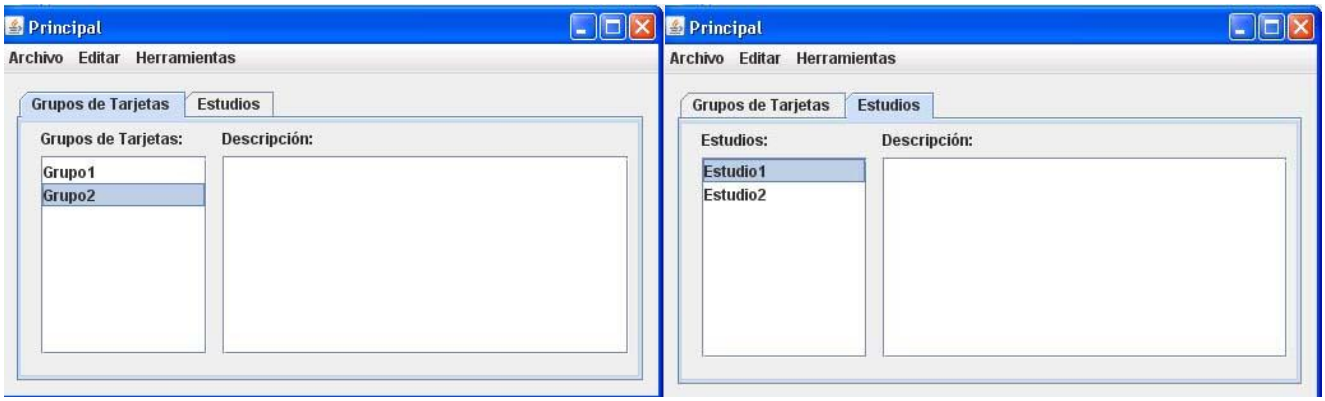
13. *Arquitectura de la información: algo más que un concepto.* **Reyes, Gómez. 2002.** La Habana : s.n., 2002. En: Congreso Internacional de Información INFO 2002.
14. *Revisión de técnicas de arquitectura de información.* **León, Rodrigo Ronda. 2007.** 1886-8592, La Habana : s.n., 5 de enero de 2007, No Solo Usabilidad.
15. *La Arquitectura de la Información y las Ciencias de la Información.* **León, Rodrigo Ronda. 2005.** 1886-8592, La Habana : s.n., 25 de Abril de 2005, No Solo Usabilidad.
16. **Martín, César. 2006.** No se habla de usabilidad. *desarrolloweb.com.* [En línea] 17 de Julio de 2006. http://www.desarrolloweb.com/manuales/5/.
17. **—. 2006.** No se habla de usabilidad. *desarrolloweb.com.* [En línea] 17 de Julio de 2006. http://www.desarrolloweb.com/manuales/5/.
18. **Yusef Hassan Montero, Francisco J. Martín Fernández. 2003.** Qué es la Accesibilidad Web. *nosolousabilidad.com.* [En línea] 14 de julio de 2003. http://www.nosolousabilidad.com/articulos/accesibilidad.htm.
19. **2007.** Navegabilidad, usabilidad y accesibilidad. [En línea] 22 de noviembre de 2007. http://teleformacion.uci.cu/file.php/33/Conferencias_Tema_VI/Conferencia_13._Navegabilidad_y_usabilidad_y_accesibilidad-_enviada_por_Wilber_22_de_nov.pdf.
20. **Gómez, n Carlos García. 2004.** *Usabilidad de las páginas de inicio de los diarios digitales españoles.* [En línea] julio-diciembre de 2004. http://ibersid.eu/ojs/index.php/scire/article/viewFile/1499/1477.
21. **G. Booch, J. Rumbaugh, I. Jacobson. 1999.** *El Lenguaje Unificado de Modelado.* s.l. : Addison Wesley Iberoamericana, 1999.
22. **Maurer, Donna. 2006.** http://www.rosenfeldmedia.com/books/cardsorting/blog/card_sort_applications/. *rosenfeldmedia.com.* [En línea] 20 de abril de 2006. http://www.rosenfeldmedia.com/books/cardsorting/blog/card_sort_applications/ .
23. **Peñalver, Gregorio Hernández. 2007.** DEPARTAMENTO DE MATEMÁTICA APLICADA. *dma.fi.upm.es.* [En línea] 8 de octubre de 2007. http://www.dma.fi.upm.es/gregorio/JavaGC/Cconvexo/teoriaJava.html.

24. http://www.webtaller.com/manual-java/origen-java.php. *webtaller.com.* [En línea] http://www.webtaller.com/manual-java/caracteristicas-java.php.

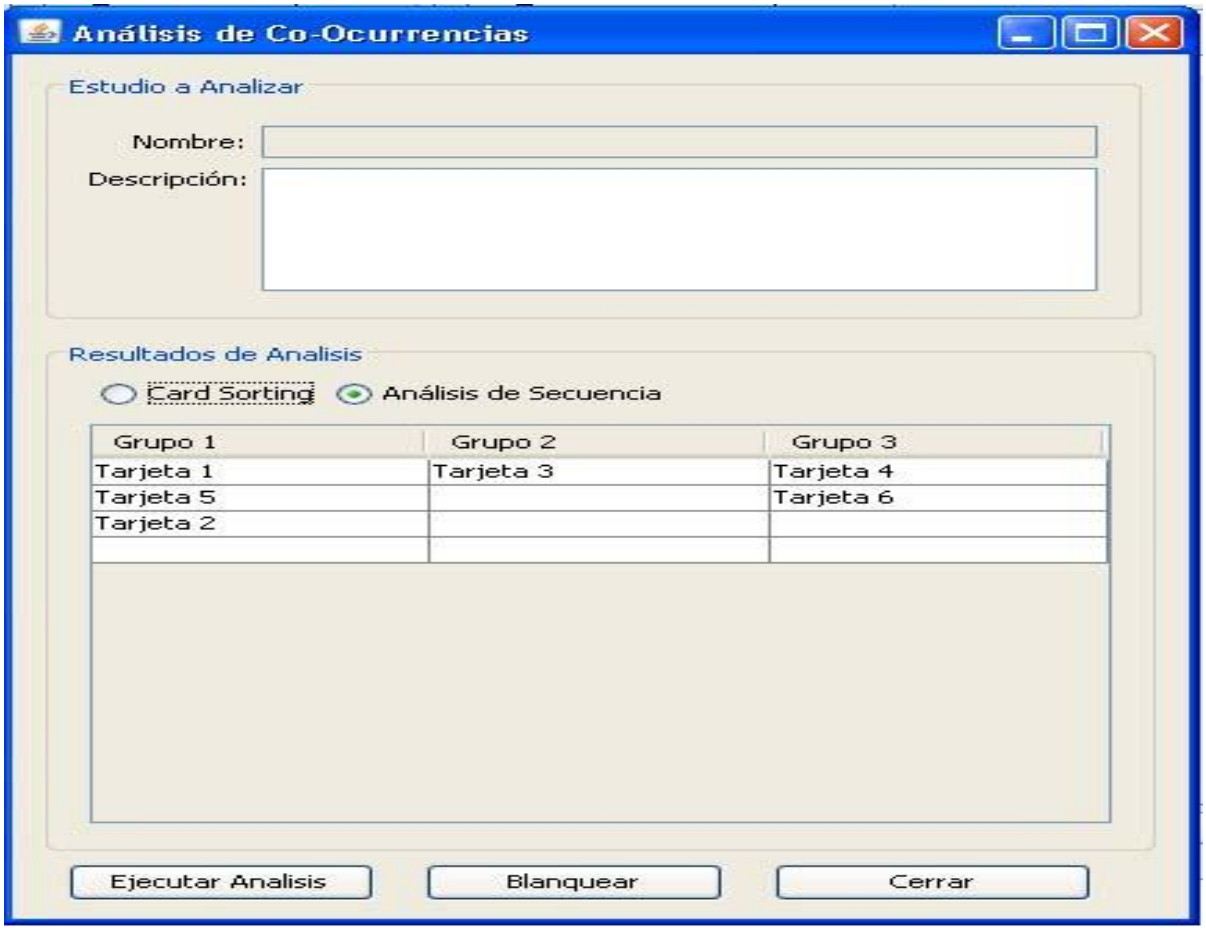
25. **Miguel Garre, Juán José Cuadrado, Miguel Ángel Sicilia.** Comparación de diferentes algoritmos de clustering en la. [En línea] http://www.sc.ehu.es/jiwdocoj/remis/docs/GarreAdis05.pdf.

ANEXOS

Anexo no. 1 Interfaz principal de la aplicación.

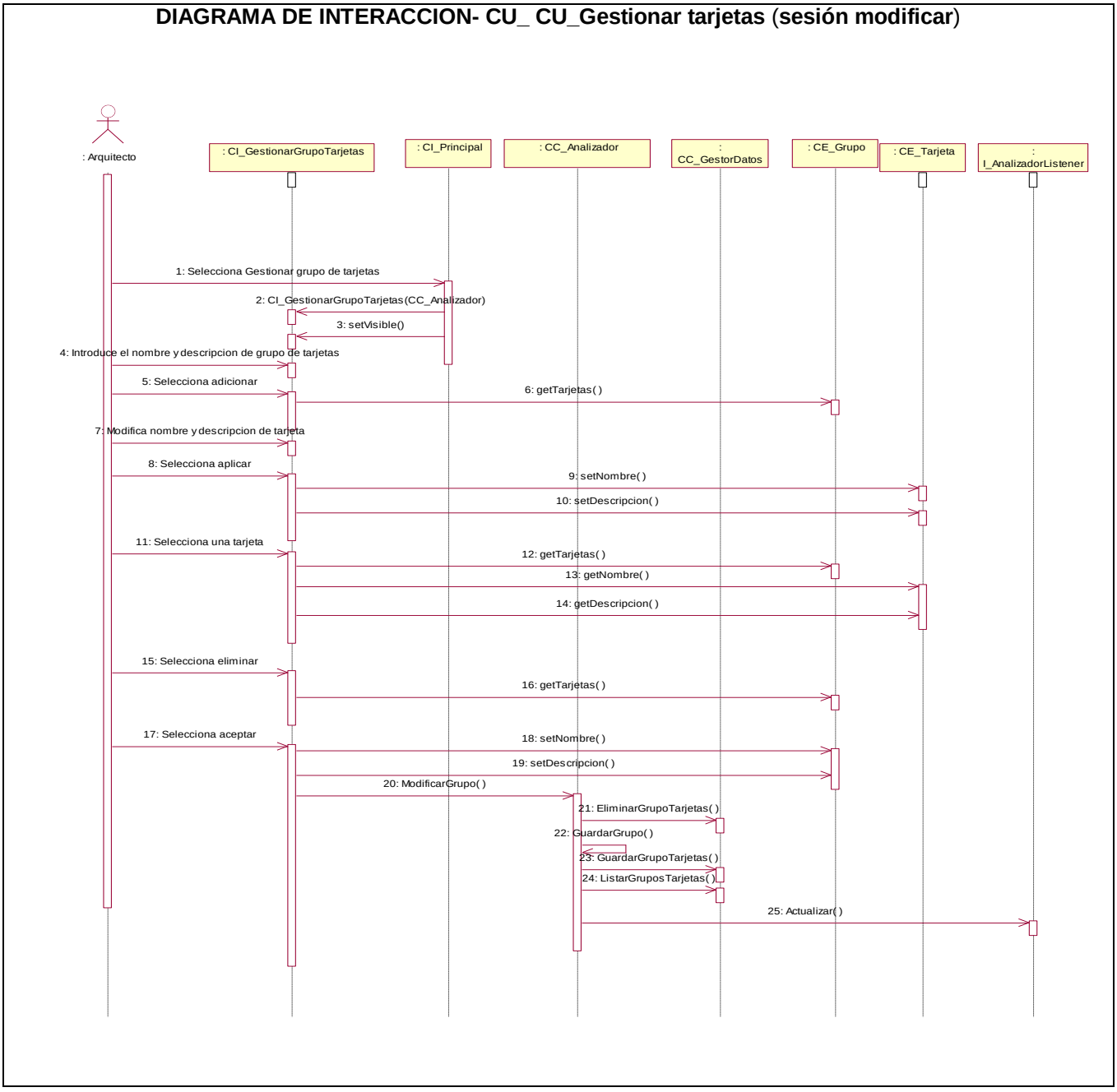


Anexo no.2 Interfaz para mostrar los resultados de los análisis realizados.



Anexo 3.

DIAGRAMA DE INTERACCION- CU_ CU_Gestionar tarjetas (sesión modificar)



Anexo 4.

DIAGRAMA DE INTERACCION- CU_Publicar estudio (sesión tipo Card Sorting)

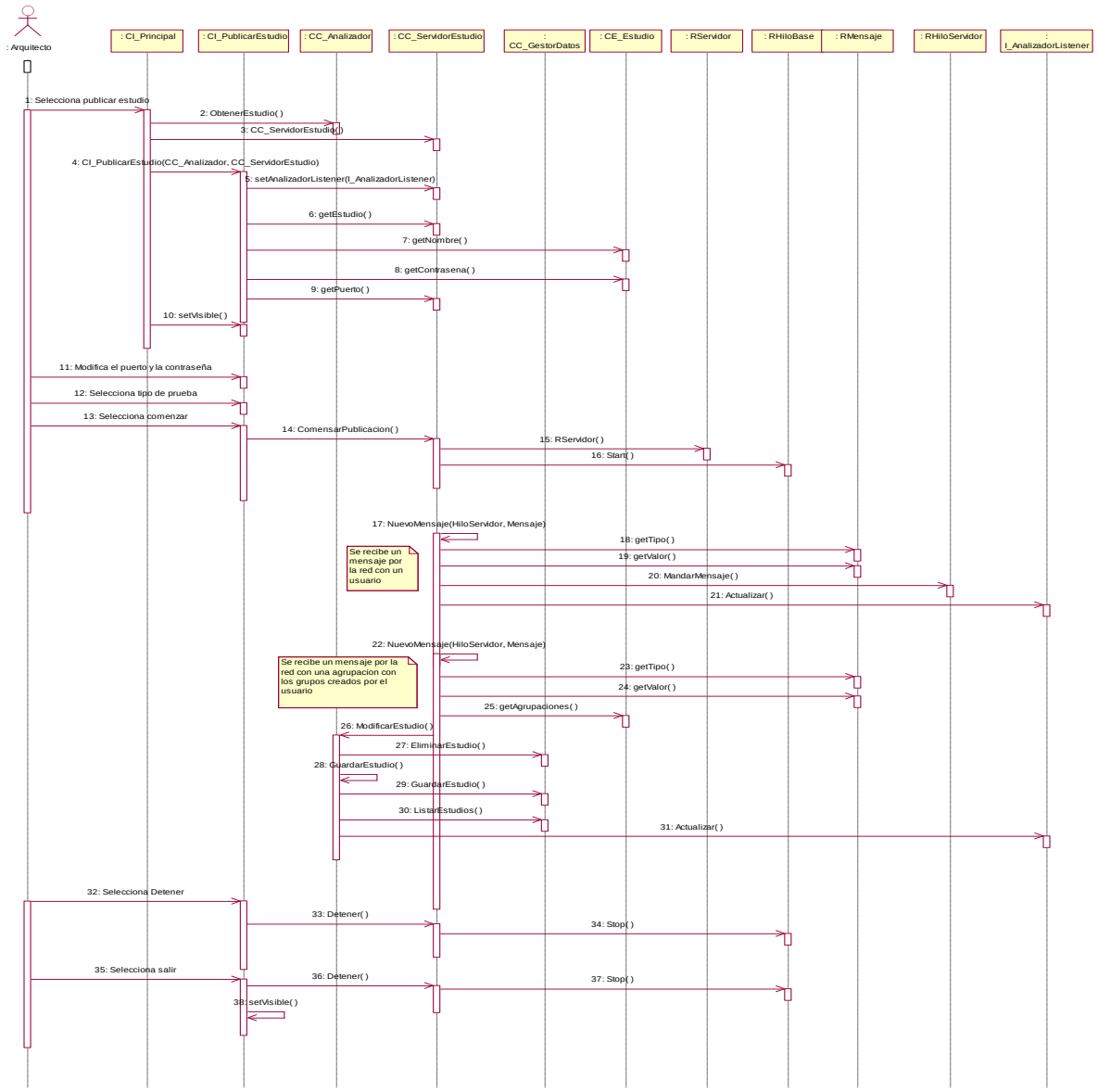


DIAGRAMA DE INTERACCION- CU_Publicar estudio (sesión tipo Análisis de Secuencia)

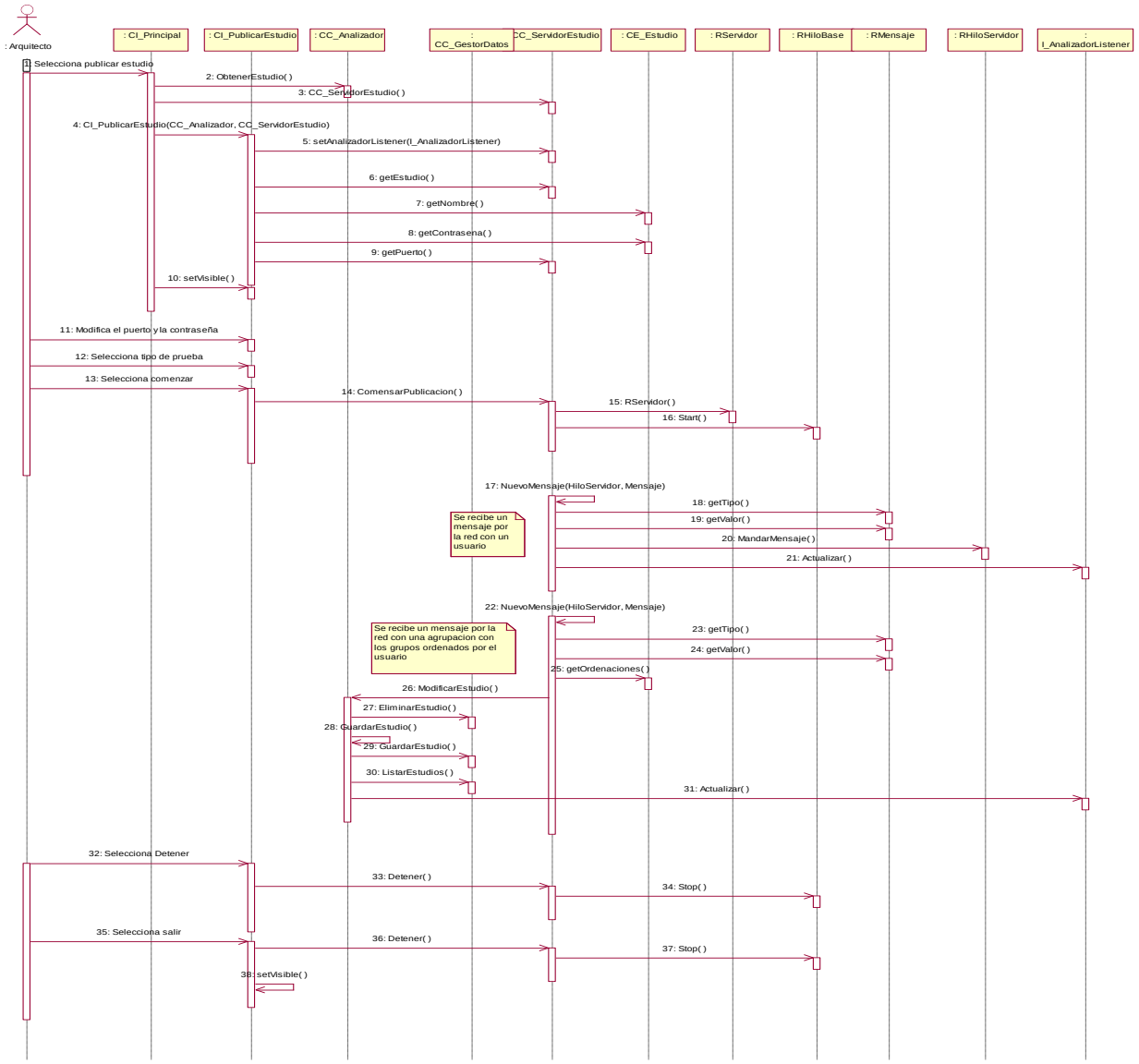
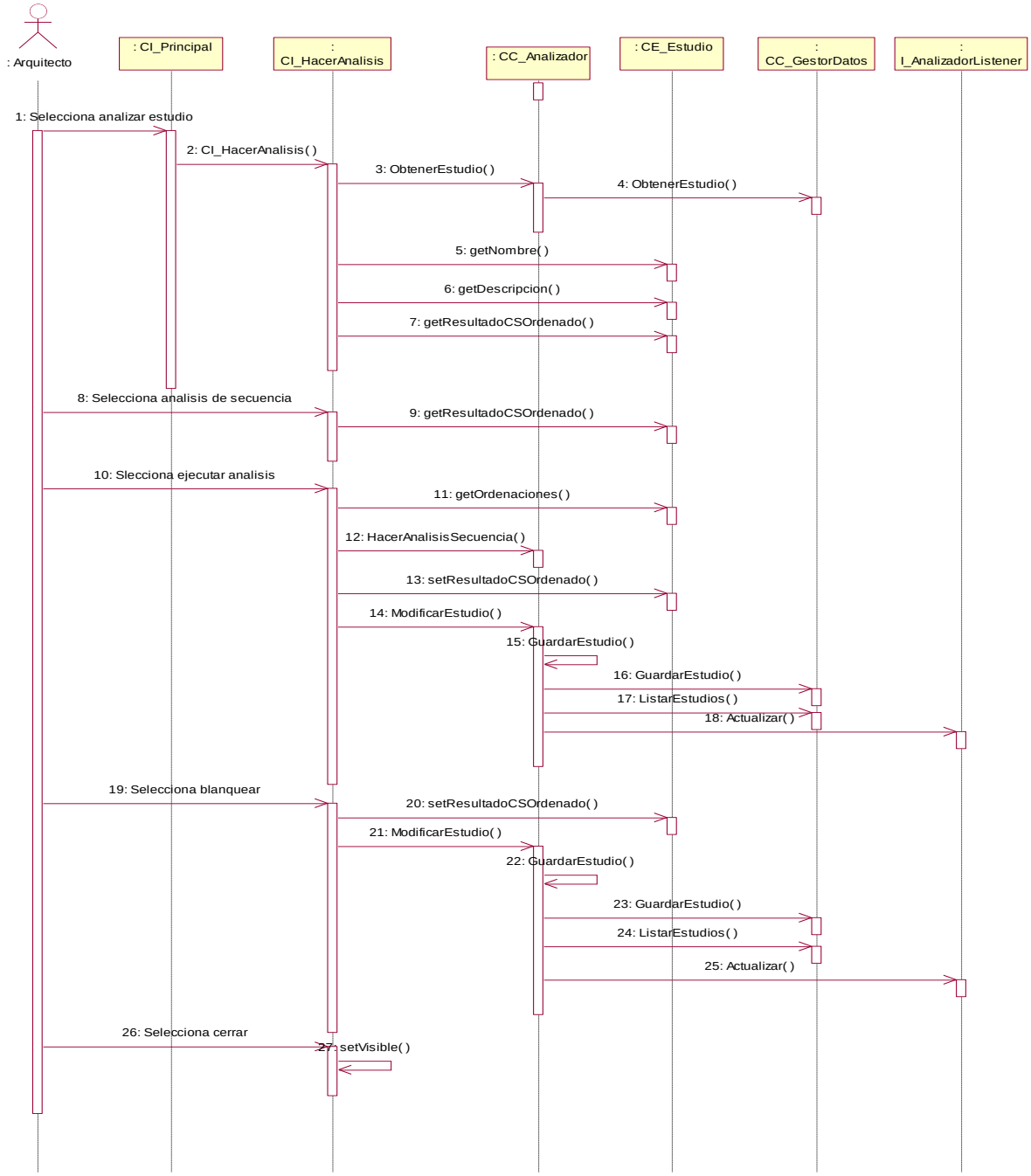


DIAGRAMA DE INTERACCION- CU_RealizarA_Coocurrencias (sesión tipo Análisis de Secuencia)



GLOSARIO

Actor: Alguien o algo, fuera del sistema o negocio que interactúa con el sistema o negocio.

Accesibilidad: Es el grado con el que algo puede ser usado, visitado o accedido por todas las personas, independientemente de sus capacidades técnicas o físicas.

API: (Application Program Interface): Interfaz de Programación de Aplicaciones es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Arquitectura de la Información (AI): Es la disciplina y arte encargada del estudio, análisis, organización, disposición y estructuración de la información en espacios de información, y de la selección y presentación de los datos en los sistemas de información interactivos y no interactivos.

Arquitectos de Información: Son las personas encargadas de poner en practicas todas las técnicas que brinda la AI.

Audiencia:

Casos de uso: Especificación de las secuencias de acciones, incluyendo secuencias variantes y secuencias de errores, que pueden ser efectuadas por un sistema, subsistema o clase por interacción con autores externos.

Card Sorting, Análisis de Secuencia: Técnicas de co-ocurrencias usadas por las arquitectos de información para la categorización de contenidos para un sitio web o multimedia.

Clases: Abstracciones que representan a un conjunto de objetos con un comportamiento e interfaz común.

Cliente. Computador que accede al servicio que brinda una PC cliente.

Componente: Una parte física reemplazable de un sistema que empaqueta su implementación, y es conforme a un conjunto de interfaces a las que proporciona su realización.

Desktop Application: Aplicación de escritorio. Aplicación que no requiere de un navegador web para ser utilizada.

DCU: Diseño centrado al usuario.

Diseñador de interfaces: Persona que diseña dispositivos de comunicación, aplicaciones de software, y sitios web enfocado en la experiencia de usuario y la interacción.

IDE: Entorno de desarrollo integrado. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un compilador, un depurador y un constructor de Interfaz Gráfica de Usuario.

Id: Identificador. Una marca que hace al artículo único en el universo de artículos.

Interfaz: Frontera convencional entre dos sistemas o dos unidades, que permite intercambio de informaciones.

Framework: Estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, y bibliotecas entre otros software.
Multimedia: término que se aplica a cualquier objeto que usa simultáneamente diferentes formas de contenido informativo como texto, sonido, imágenes, animación y video para informar o entretener al usuario.

Plataforma de desarrollo: Es el entorno común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo.

Proyecto Web:

Requisito o Requerimiento: Una característica, propiedad o comportamiento que se desea para el sistema.

RUP: El Proceso Unificado Racional o RUP (Rational Unified Process), es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Servidor: Es un computador que provee una clase especial de servicio a los software clientes que están corriendo en otros computadores y que lo accedan para realizar una función determinada.

Sistema: Colección de unidades conectadas que se organiza para lograr un propósito. El sistema es el "modelo completo".

Sistema de ejecución Run-time: Sistema que ejecutan programas de computadora en un determinado intervalo de tiempo Este tiempo se inicia con la puesta en memoria principal del programa y finaliza en el momento en que éste envía al sistema operativo la señal de término.

Software:

TCP/IP: Siglas que significan "*Transmisión Control Protocol*" e "*Internetwork Protocol*" o protocolo de trabajo en internet. Estos dos nombre compuestos son el llamado estándar de comunicaciones para intercomunicar las diversas redes que existen.

UML: es el Lenguaje de Modelación Unificado es un lenguaje gráfico para detallar, construir, visualizar y documentar las partes o artefactos (información que se utiliza o produce mediante un proceso de software). Pueden ser artefactos: un modelo, una descripción que comprende el desarrollo de software que se basen en el enfoque Orientado a Objetos, utilizándose también en el diseño de multimedia.

Usabilidad: Medida empírica y relativa de la usabilidad de un sistema.

Usuarios: Es la persona que utiliza o trabaja con algún objeto o que es destinaría de algún servicio público o privado, empresarial o profesional.