***Modelo de Diseño***

**Proyecto: <Convenios Institucionales>**

**Versión: <1.0>**

**Historial de Revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Autor** | **Descripción** |
| *<x.y.z>* | *<dd/mm/aa>* | *<nombre>* | *<especificaciones>* |

**TABLA DE CONTENIDO**

1. **Introducción**
   1. Alcance

En este documento podremos ver el contenido correspondiente al diseño de la aplicación web del protocolo de la Universidad Francisco de Paula Santander, así como visualizar la estructura y comportamiento del modelo UML de la aplicación y otros elementos clave en su diseño e implementación

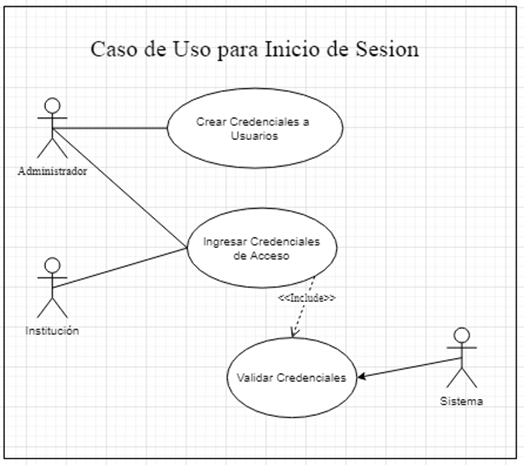
* 1. Definiciones, acrónimos y abreviaturas

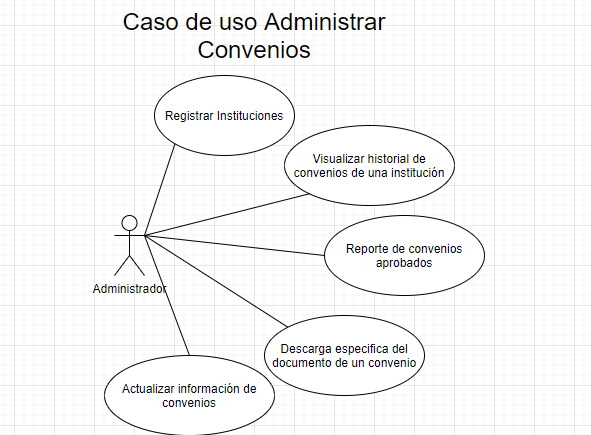
GUI: abreviatura que determina que este objeto es una interfaz del sistema.

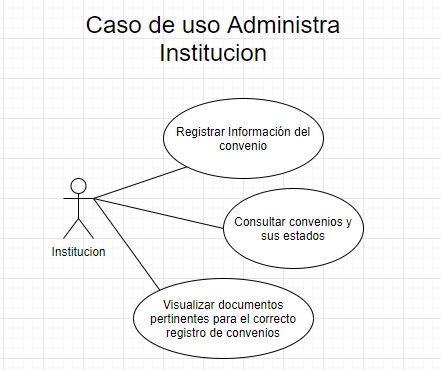
Base de datos: Estas bases de datos modifican la información almacenada a lo largo del tiempo, además de las operaciones básicas de consulta, también permiten operaciones como la actualización de datos. Ejemplos de esto podrían ser las bases de datos que se utilizan en supermercados, farmacias, tiendas de videos o sistemas de información de la empresa

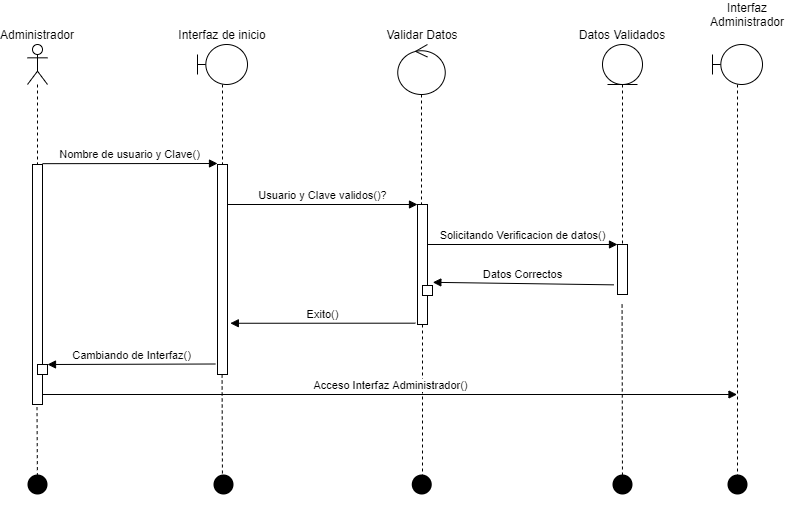
* 1. Documentos relacionados y Referencias

Se hace referencia a los documento de Visión, de Casos de Uso y de Arquitectura del sistema, entre otros.

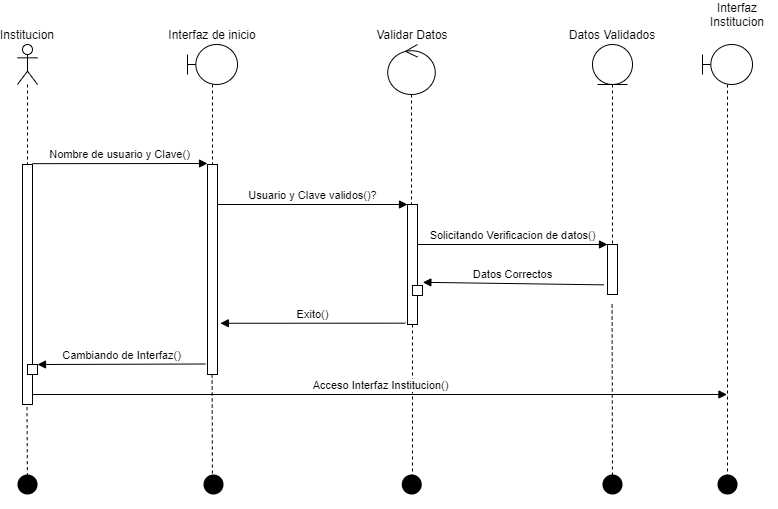
****

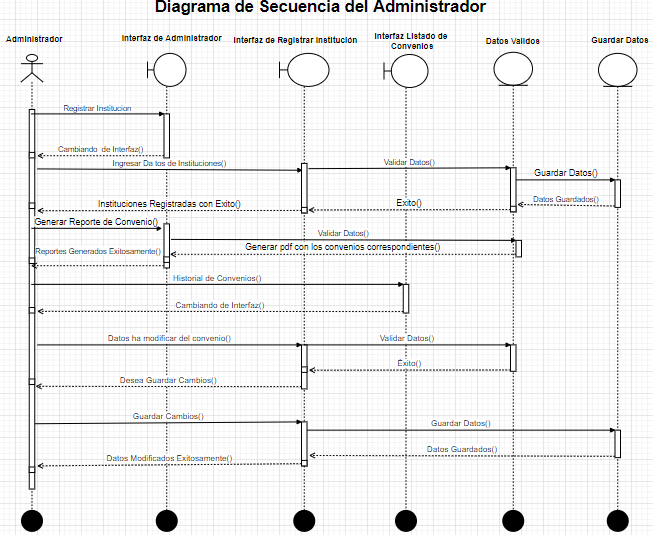


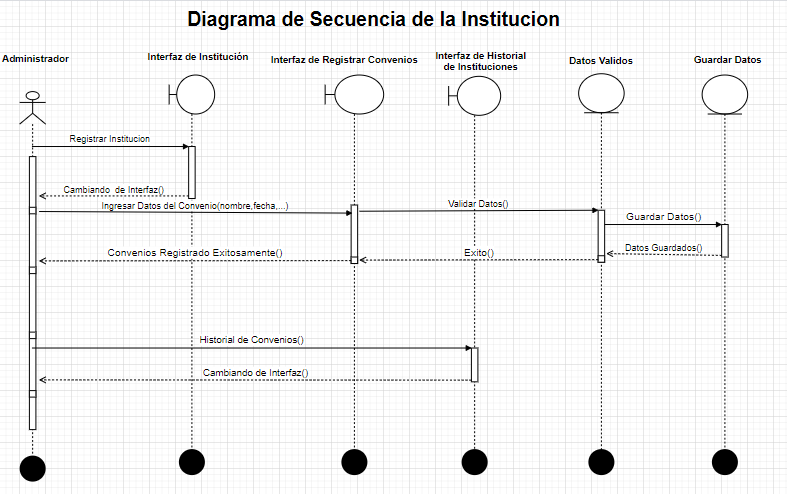
****

**Diagramas de secuencia de inicio de sesión del administrador.**

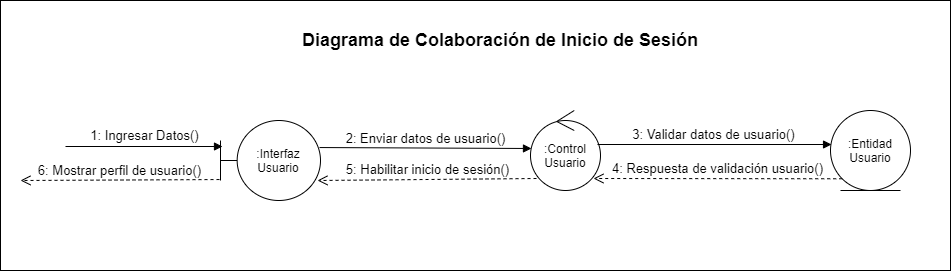
**Diagramas de secuencia de inicio de sesión de la institución.**

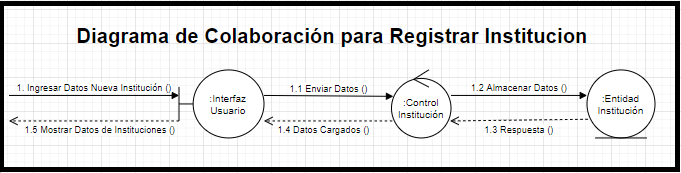


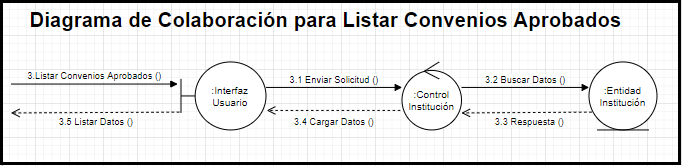
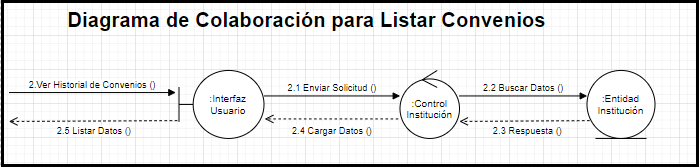


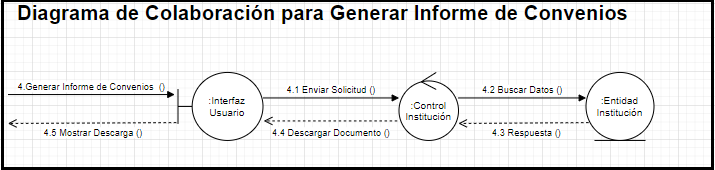


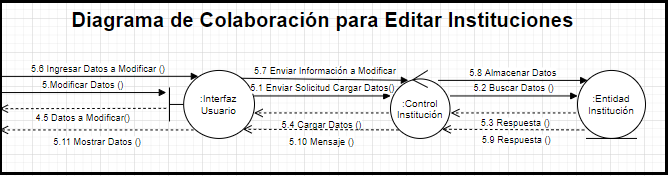
**Diagramas de Colaboración**

**

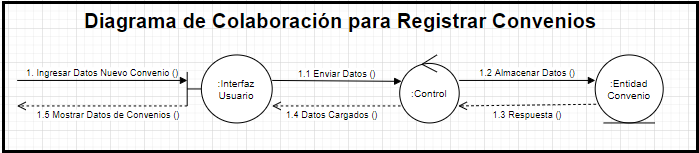
**

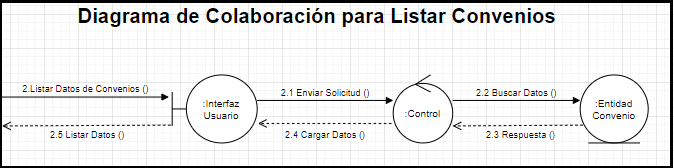
**

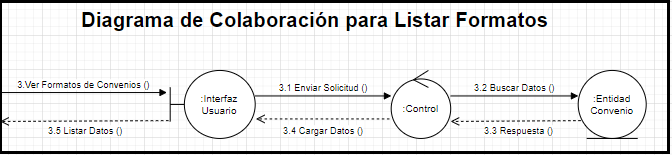
**

**

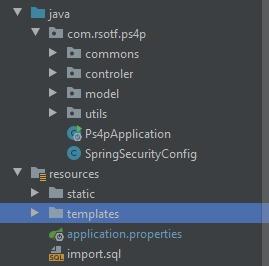
**Diagramas de colaboración de la institución.**

****

**

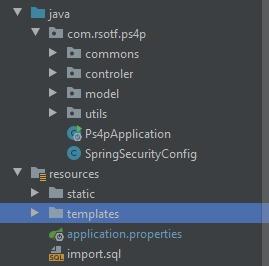
**

1. Arquitectura del software actual



1. Arquitectura del software propuesto.

MVC (MODELO – VISTA – CONTROLADOR)



* 1. Metas arquitectura y Restricciones

**Performance:** El desempeño de la aplicación debe ser muy eficiente de tal manera el usuario inmediato y todos los demás observen lo más rápidamente posible los cambios realizados en un momento determinado.

**Eficiente:** Aumentar la eficiencia de los sistemas al introducir procesos que permitan medir mediante normas específicas, la calidad del *software* desarrollado, buscando siempre la mejor calidad posible según las necesidades y resultados que se quieren generar.

**Adaptabilidad:** Mejorar el diseño de aplicaciones o *software* de tal modo que se adapten de mejor manera a las necesidades de las organizaciones o finalidades para las cuales fueron creadas.

**Usabilidad:** El diseño debe ser orientado por y para la comodidad del usuario, de manera que la interfaz sea intuitiva y fácil de manejar, al mismo tiempo que se fomente altamente la interacción entre ambos. De la misma forma, el usuario debe tener la capacidad de equivocarse y regresar a un estado seguro en el que se le permita cumplir con su objetivo original sin que se le haga tedioso o complicado el proceso para llegar a dicho fin.

En la planificación de este proceso hemos encontrado las siguientes restricciones:

1. Restricciones de contenido: debido a que nuestro sistema está basado y propuesto para aquellos que pertenecen a la institución educativa, el mismo debe estar de acuerdo con el reglamento de la institución, por lo que ciertos contenidos deben ser excluidos en el uso de la aplicación.
2. Restricciones de tecnología y uso de herramientas de desarrollo: están predefinidos los instrumentos a utilizar, así como también la plataforma tecnológica sobre la que se va a desarrollar el sistema. La aplicación será implementada usando Java Framework Spring Boot, Ocirep, JavaScript, es por ello que las herramientas utilizadas estarán determinadas por las funcionalidades ofrecidas por dicho lenguaje.
   1. Vista de Casos de Uso

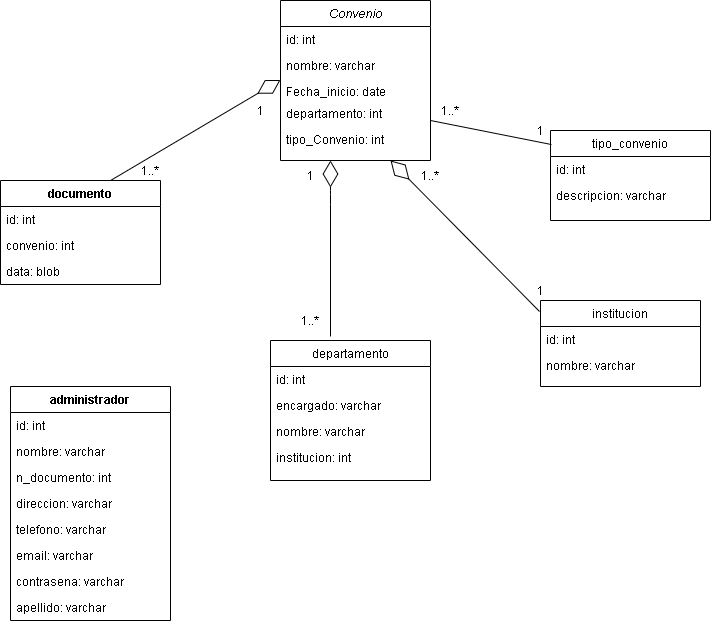
A cada uno se le hará una descripción en formato breve para enunciar su Escenario Principal de Éxito. Se utilizará el Diagrama de Casos de Uso en notación UML

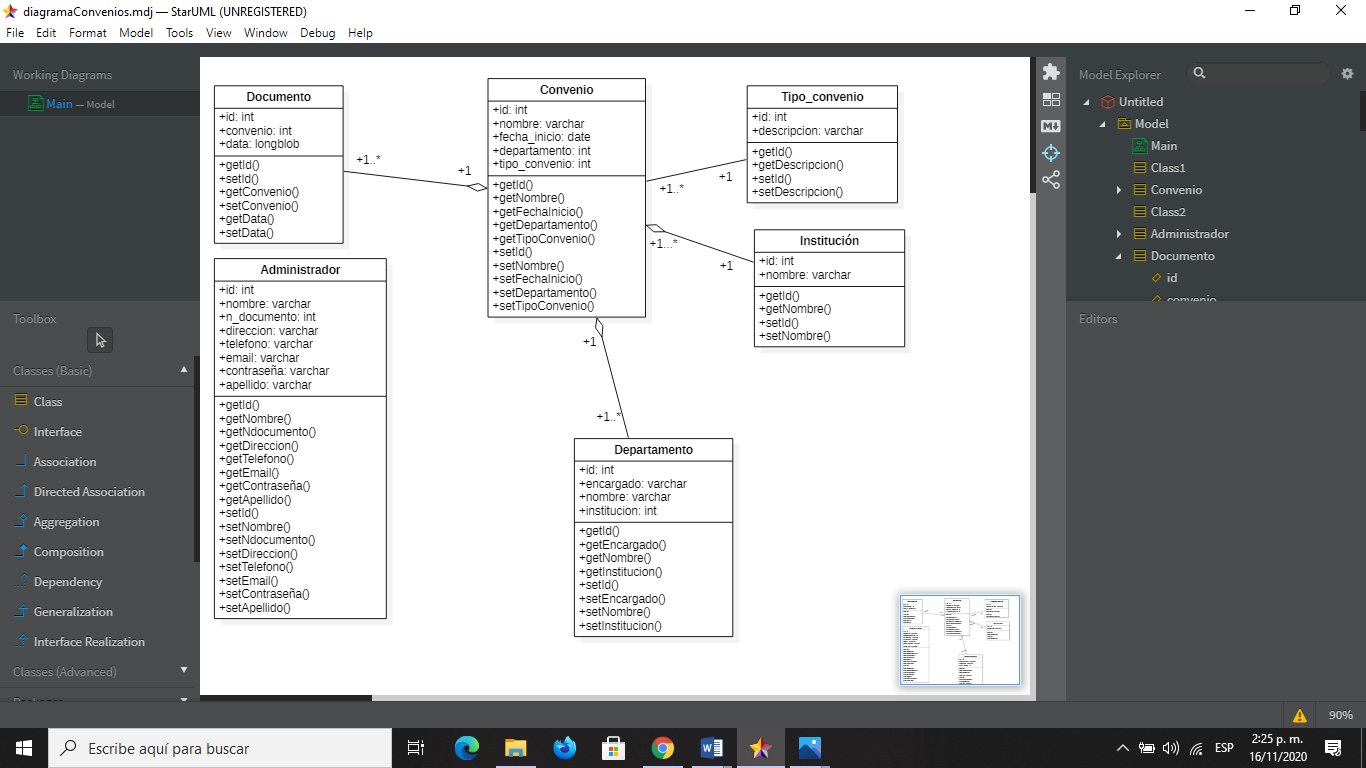
|  |  |  |
| --- | --- | --- |
| **Caso de Uso*-****CDU-001* | | |
| **Código CU: 001** | **Prioridad:** | |
| **Nombre:** | Iniciar sesión. | |
| **Descripción:** | *Para iniciar sesión se debe ingresar al aplicativo web en el cual se ingresarán los datos siguientes: Usuario y contraseña.* | |
| **Requerimiento:** | *RF1 - Iniciar sesión*  *RF - Recuperar contraseña*  *RF - Validar Usuarios* | |
| **Precondición:** | *Los usuarios deben estar previamente registrados en el sistema.* | |
| **Flujo Normal:** | | |
| *El sistema debe capturar los datos (usuario y la contraseña) digitalizados por el usuario y validar que estos datos se encuentren almacenados en la base datos y así verificar que esté registrado, de esta forma se le permite al usuario iniciar sesión.* | | |
| **Actor** | | **Sistema** |
| 1. *El usuario accede al aplicativo web.*  3. *El usuario ingresa el usuario y la contraseña.*  6. *El usuario inicia sesión.* | | 2. *El sistema muestra un formulario con los datos necesarios para el inicio de sesión.*  4. *El sistema procede a realizar la autentificación de los datos.*  5. *El sistema valida que el tipo de usuario.*  7. *Se redirige a la página de Convenios o panel de instituciones registradas , esto depende del tipo de usuario .* |
| **Flujo Alterno:** | | |
| *La contraseña es incorrecta.* | | |
| **Actor** | | **Sistema** |
| 1.1. *El usuario accede al aplicativo web.*  1.3. *El usuario ingresa el usuario y la contraseña.*  1.8. *El usuario no puede iniciar sesión.*  2.1. *El usuario ingresa el usuario y la contraseña.*  2.3. *El usuario elige recuperar contraseña.*  2.5. *El usuario ingresa el correo para recuperar o cambiar su clave.*  2.7. *El usuario obtiene una nueva contraseña. el usuario ingresa sus datos nuevamente y selecciona el rol*  2.9. *El usuario escoge el rol para ingresar al sistema.*  2.11. *El usuario inicia sesión.*  3.1. *El usuario ingresa Usuario que no se encuentran registrados*  3.3. *El usuario no puede entrar al sistema*  4.1. *El usuario ingresa su Usuario y Contraseña*  4.3. *El usuario elige recuperar contraseña*  4.5. *El usuario digita un correo de recuperación incorrecto o no correspondiente*  4.7. *El usuario no obtiene la nueva contraseña* | | 1.2. *El sistema muestra un formulario con los datos necesarios para el inicio de sesión.*  1.4. *El sistema procede a realizar la autentificación de los datos.*  1.5. *El sistema valida que el tipo de usuario.*  1.6. *El sistema detectó que la contraseña es incorrecta.*  1.7. *El sistema notifica al usuario sobre lo sucedido.*  2.2. *El sistema valida que la contraseña es incorrecta.*  2.4. *El sistema le pide digitar el correo registrado para la recuperación de la contraseña.*  2.6. *El sistema valida que el correo sea el que se encuentra en la base de datos, genera una nueva contraseña, actualiza la contraseña en la base de datos y envía la nueva contraseña a dicho correo.*  2.8. *El sistema valida que el usuario, código y contraseña digitados estén en la base de datos.*  2.10. *El sistema valida que el rol seleccionado sea el correspondiente al usuario*  3.2. *El sistema valida que los datos ingresados no se encuentren en la base de datos y muestra un mensaje donde se notifica que el usuario no existe.*  4.2. *El sistema valida que la contraseña es incorrecta.*  4.4. *El sistema le pide ingresar el correo registrado para la recuperar la contraseña.*  4.6. *El sistema valida que el correo no se encuentra registrado en la base de datos del sistema.* |
| **Postcondición:** | | *Permite el acceso de los usuarios al sistema mostrando su respectiva interfaz de usuario* |
| **Requerimientos Especiales:** | |  |
| **Puntos de Extensión:** | |  |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso*-****CDU-002* | | |
| **Código CU: 002** | **Prioridad:** | |
| **Nombre:** | *Registrar instituciones.* | |
| **Descripción:** | *El sistema permite al administrador registrar una institución ingresando la información: Nombre de la institución.* | |
| **Requerimiento:** | *RF2 - Registrar Instituciones.* | |
| **Precondición:** | *El nombre de la institución no deben encontrarse registrados en la base de datos del sistema.* | |
| **Flujo Normal:** | | |
| *El sistema debe capturar los datos (Nombre de la institución.) digitalizados por el usuario y validar que los datos (Nombre de la institución ) no se encuentren almacenados en la base datos y así verificar que es un nuevo usuario, de esta forma se le permite al usuario registrarse.* | | |
| **Actor** | | **Sistema** |
| 1. *El administrador accede a registrar la institución.*  3. *El administrador ingresa todos los datos requeridos en el formulario.*  5. *La institución es registrado en el sistema.* | | 2. *El sistema muestra un formulario con los datos que se deben completar para el registro de la institución.*  4. *El sistema valida que el nombre no se encuentre almacenados en la base de datos del aplicativo web.* |
| **Flujo Alterno:** | | |
| *El nombre de la institución ya se encuentran registrados.* | | |
| **Actor** | | **Sistema** |
| * 1. *El administrador accede a registrar la institución .*   1.3. *El administrador ingresa todos los datos requeridos en el formulario.*  1.5. *El administrador no puede registrar usuario.* | | 1.2. *El sistema muestra un formulario con los datos que se deben completar para el registro de la institución.*  1.4. *El sistema valida que el nombre de la institución se encuentren almacenados en la base de datos del aplicativo web.*  1.6. *El sistema notifica al administrador que los datos ya se encuentran registrados en la base de datos.* |
| **Postcondición:** | | *Permite a la institución ver su interfaz de bienvenida.* |
| **Requerimientos Especiales:** | |  |
| **Puntos de Extensión:** | |  |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso*-****CDU-002* | | |
| **Código CU: 002** | **Prioridad:** | |
| **Nombre:** | *Visualizar Historial de Convenios de una institución* | |
| **Descripción:** | *El sistema permite al administrador visualizar los convenios de una institución ingresando la información: Nombre de la institución.* | |
| **Requerimiento:** | *RF2 - Visualizar Historial de Convenios de una institución* | |
| **Precondición:** | *El nombre de la institución deben encontrarse registrados en la base de datos del sistema.* | |
| **Flujo Normal:** | | |
| *El sistema debe capturar los datos (Nombre de la institución.) digitalizados por el usuario y validar que los datos (Nombre de la institución ) se encuentren almacenados en la base datos y así verificar si existen convenios con esa institución .* | | |
| **Actor** | | **Sistema** |
| 1. El administrador accede a buscar la institución con sus respectivos convenios  4. *El administrador obtiene el historial de convenio* | | 2. *El sistema muestra un campo para que el administrador digite el nombre de la institución.*  3. *El sistema valida que el nombre y tenga registrado algún convenio con esa institución se encuentre almacenados en la base de datos del aplicativo web.* |
| **Flujo Alterno:** | | |
| *El nombre de la institución no tiene convenios.* | | |
| **Actor** | | **Sistema** |
| * 1. El administrador accede a buscar la institución con sus respectivos convenios   1.5. *El administrador no recibe ningún resultado de la busqueda ya que la institución no existe.* | | 1.2.2. *El sistema muestra un campo para que el administrador digite el nombre de la institución.*  1.4. *El sistema valida que el nombre de la institución se encuentren almacenados en la base de datos del aplicativo web.*  1.6. *El sistema notifica al administrador que los datos no se encuentran registrados en la base de datos.* |
| **Postcondición:** | | *Permite a la institucion ver su interfaz de bienvenida.* |
| **Requerimientos Especiales:** | |  |
| **Puntos de Extensión:** | |  |

* 1. Vista lógica del Sistema



* + 1. Modelo conceptual (diagrama de clases) 

Clase Documento: guarda la información respecto a que requisitos piden para establecer un Convenio.

Clase Convenio: está clase se encarga de guardar el convenio que tiene con cada departamento de una institución.

Clase Departamento: está guarda los datos del departamento que tiene cada institución.

Clase Tipo Convenio: como su nombre lo indica el tipo de convenio guarda la información de qué convenio es, con su respectiva descripción.

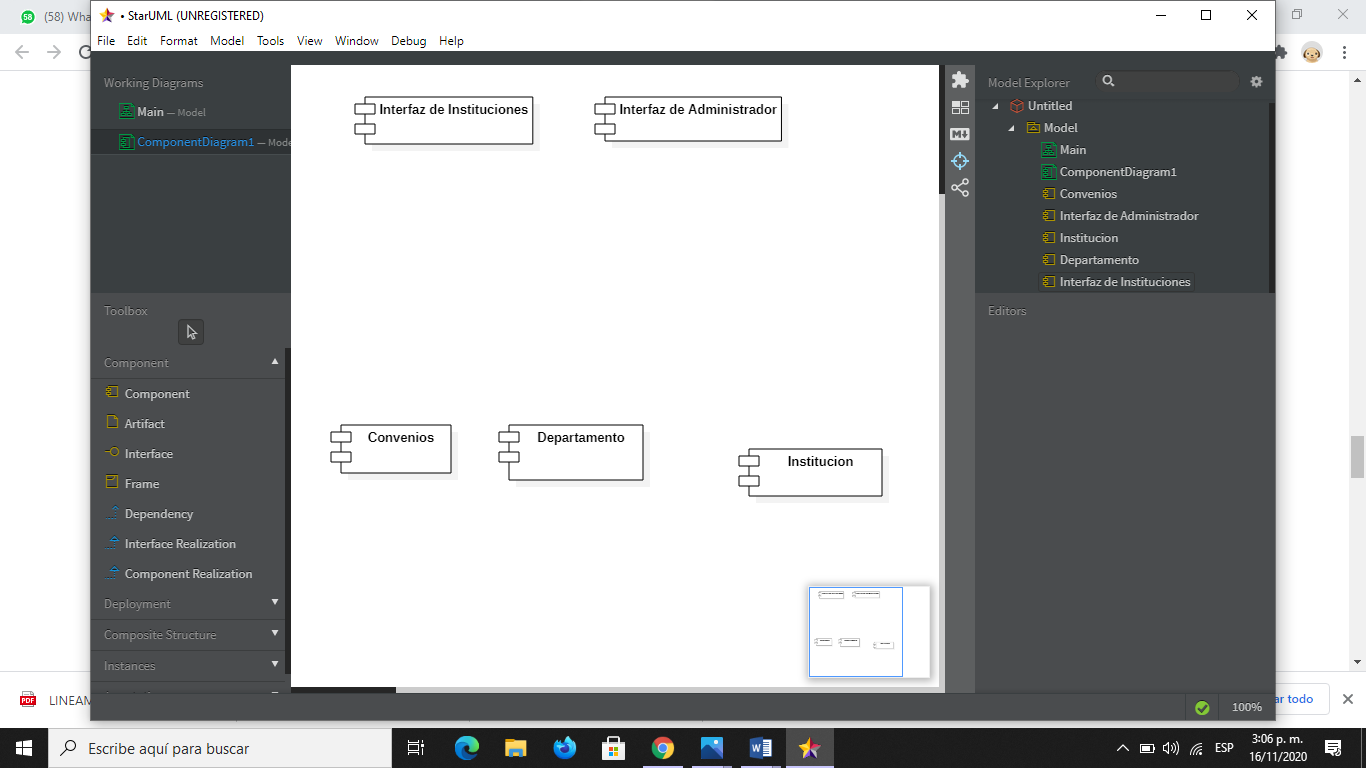
Clase Institución: está guarda la información de la institución a la cual se le desea hacer un convenio.

Clase Administrador: que es el que registra las intituciones en el sistema.

* + 1. Diagrama de componentes

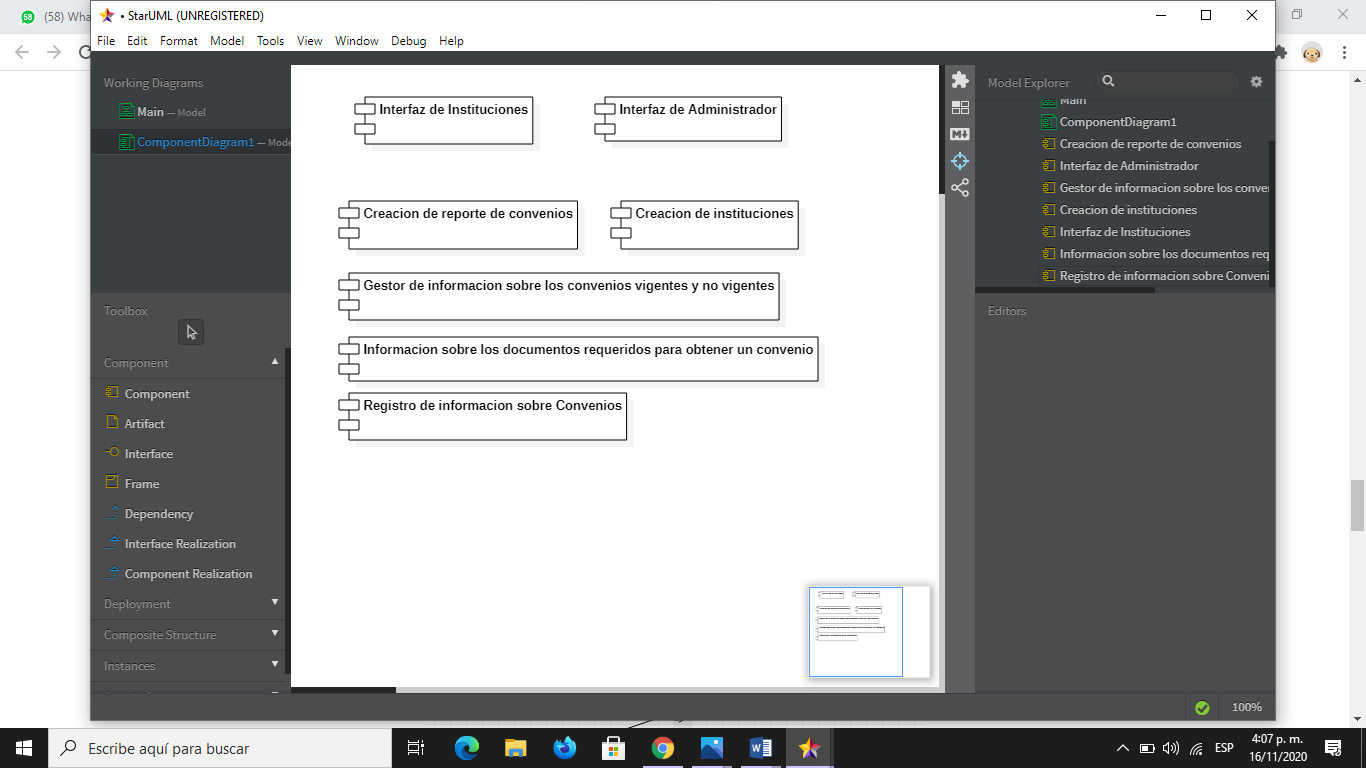


















* + 1. Diagrama de paquetes









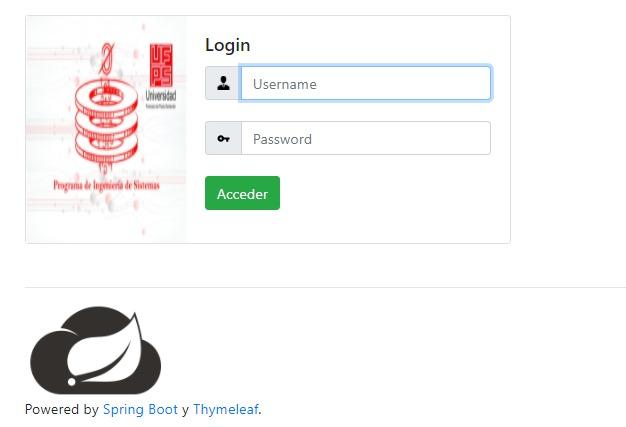


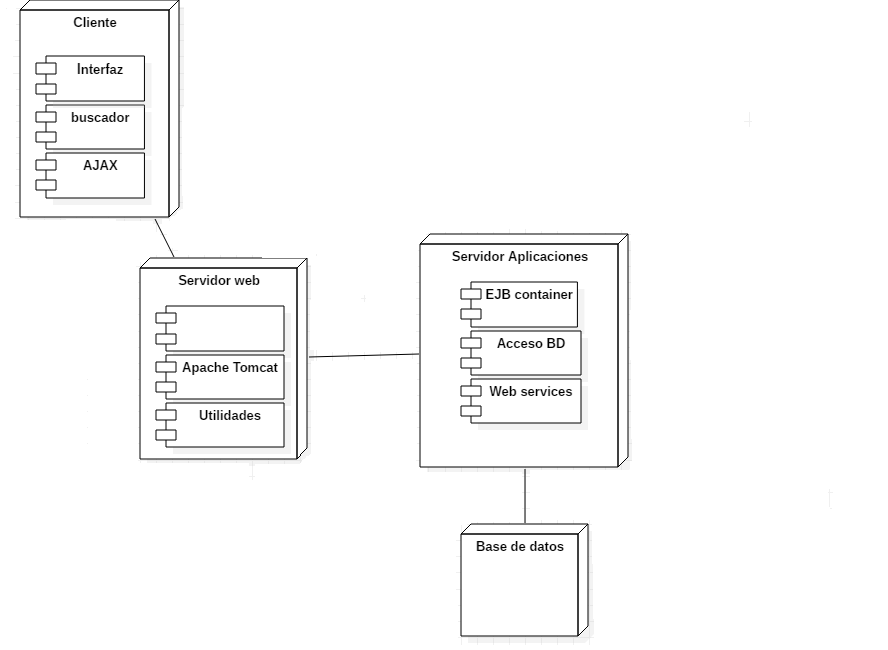








* 1. Vista física del Sistema 
     1. Diagrama de despliegue



* 1. Vista de Implementación

Para el desarrollo del aplicativo web se usó la arquitectura MVC, en la cual tiene los archivos responsables del funcionamiento del aplicativo dividido en paquetes diferentes manteniendo separadas las vistas, el modelo y el controlador, otro beneficio es esta **tecnología se centra en la escalabilidad**, y permite a su vez dividir el trabajo entre un grupo de profesionales, al estar sus componentes separados entre sí

La arquitectura MVC propone, independientemente de las tecnologías o entornos en los que se base el sistema a desarrollar, la **separación de los componentes de una aplicación en tres grupos** (o capas) principales: **el modelo, la vista, y el controlador**, y describe **cómo se relacionarán entre ellos** para mantener una estructura organizada, limpia y con un acoplamiento mínimo entre las distintas capas.

* + 1. Visión General

La aplicación web de convenios institucionales de desarrollo de software tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente.

Es actividades requeridas para desarrollar el sistema de software de alta calidad y proporciona el marco de trabajo desde el cual se puede establecer un plan detallado para el desarrollo del software. Actividades: Diseño, validación, evolución, especificación.

* + 1. Capas.

El **Modelo**: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación ([lógica de negocio](https://es.wikipedia.org/wiki/L%C3%B3gica_de_negocio)). Envía a la vista aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al modelo a través del controlador.

El **Controlador**: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al modelo cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su vista asociada si se solicita un cambio en la forma en que se presenta el modelo (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el controlador hace de intermediario entre la vista y el modelo .

La **Vista**: Presenta el 'modelo' (información y *lógica de negocio*) en un formato adecuado para interactuar (usualmente la [interfaz de usuario](https://es.wikipedia.org/wiki/Interfaz_de_usuario)), por tanto requiere de dicho 'modelo' la información que debe representar como salida.

* + - 1. Capa de Interfaz

|  |  |
| --- | --- |
| **Área** | **Productos / Servicios / Componentes** |
| Estilo de Layout y Usabilidad | 1. El layout que se presentará para la aplicación thymeleaf presentará un estilo consistente y será amigable al usuario, de manera que se facilite la interacción. 2. La aplicación se utilizará a través del sistema thymeleaf mediante un navegador Web o Browser. |
| **Herramientas de Construcción:** |  |
| Lenguajes | 1. JAVA. Es imprescindible para la generación de la interfaz en las páginas de contenido dinámico. |
| Despliegue de Información | 1. Hypertext Markup Language (HTML) se empleará para transmitir la información al usuario. |
| **Componentes:** |  |
| Presentación de errores a la interfaz del usuario | 1. Se mostrarán los errores en la página comunicándole al usuario la falla ocurrida de una manera entendible para que pueda ser comunicada fácilmente a un administrador. |

* + - 1. Capa de Negocio

La definición de los límites de cada capa nos permitirá definir correctamente la colaboración que proporcionará cada una de ellas y descubriremos que la capa intermedia es inevitablemente la lógica de negocios. Esto dará lugar a una infraestructura robusta y lista para la extensión y el crecimiento como proveedora de servicios.

Spring es un framework open source que proporciona un marco de trabajo para el desarrollo de aplicaciones J2EE. El framework está basado en el uso de ficheros planos JavaBeans para la lógica de aplicación y archivos XML para la configuración.

* + - 1. Capa de Datos

la persistencia se logra por serialización del objeto o almacenamiento en una base de datos. Las bases de datos más populares hoy en día son relacionales.

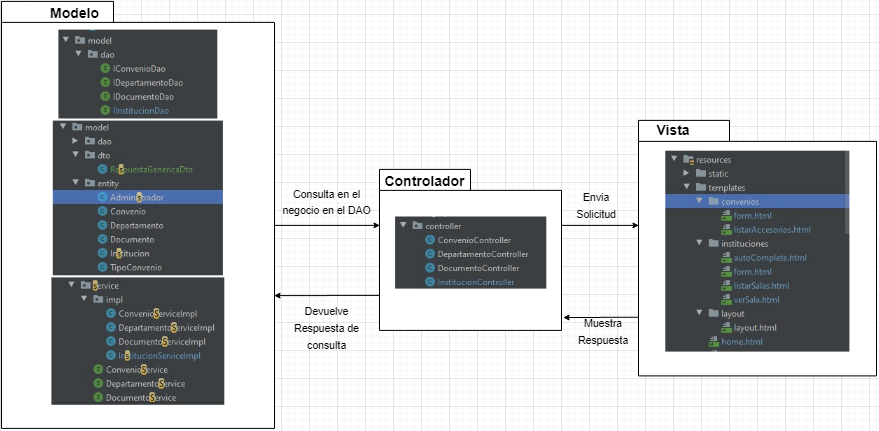
El modelo de objetos difiere en muchos aspectos del modelo relacional. La interfaz que une esos dos modelos se llama asociación objeto-relacional (ORM en inglés). Una capa de persistencia encapsula el comportamiento necesario para mantener los objetos. O sea: leer, escribir y borrar objetos en el almacenamiento persistente (base de datos). La persistencia de la información es la parte más crítica en una aplicación de software.

Se debe permitir que varios usuarios trabajen en la misma base de datos, protegiendo los datos de ser escritos erróneamente. Para ello, se utilizará el bloqueo optimista, soportado por la mayoría de los motores de persistencia como la opción por defecto, ya que, en la mayoría de los casos, es suficiente.

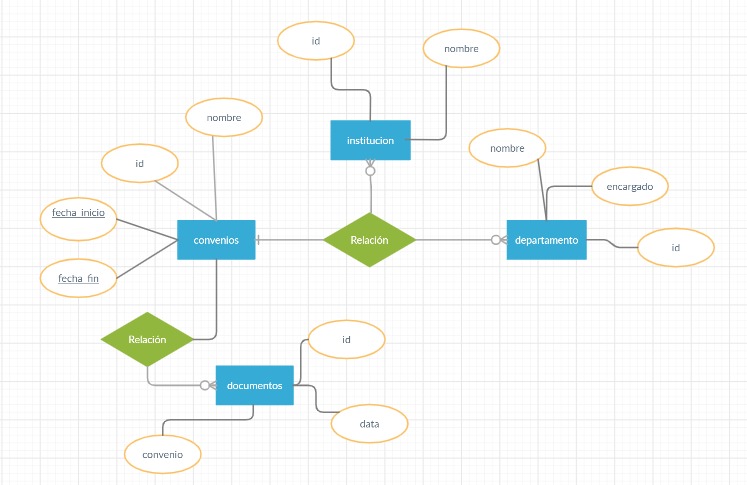
* 1. Vista de Procesos

Se habla de los procesos (si existen).

1. Diseño Detallado
   1. Estructura del Sistema

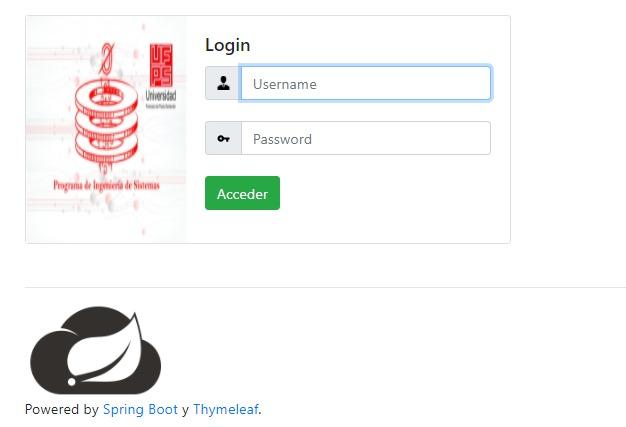


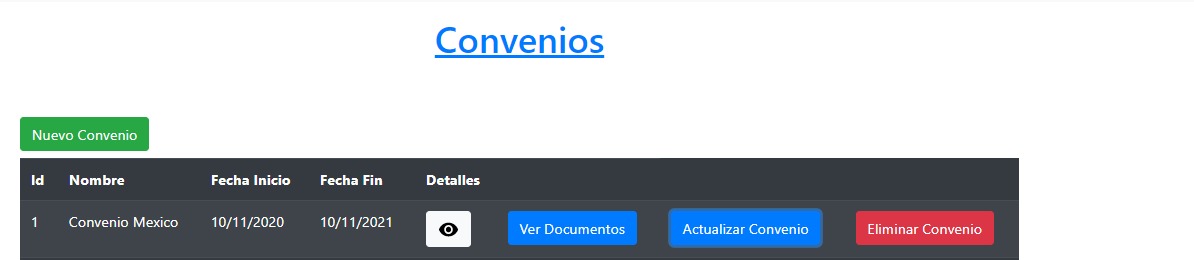
* + 1. Modelo Entidad Relación

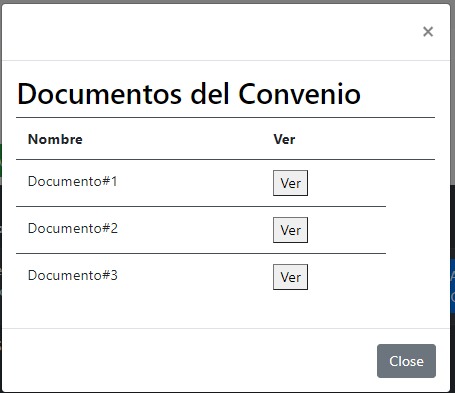
****

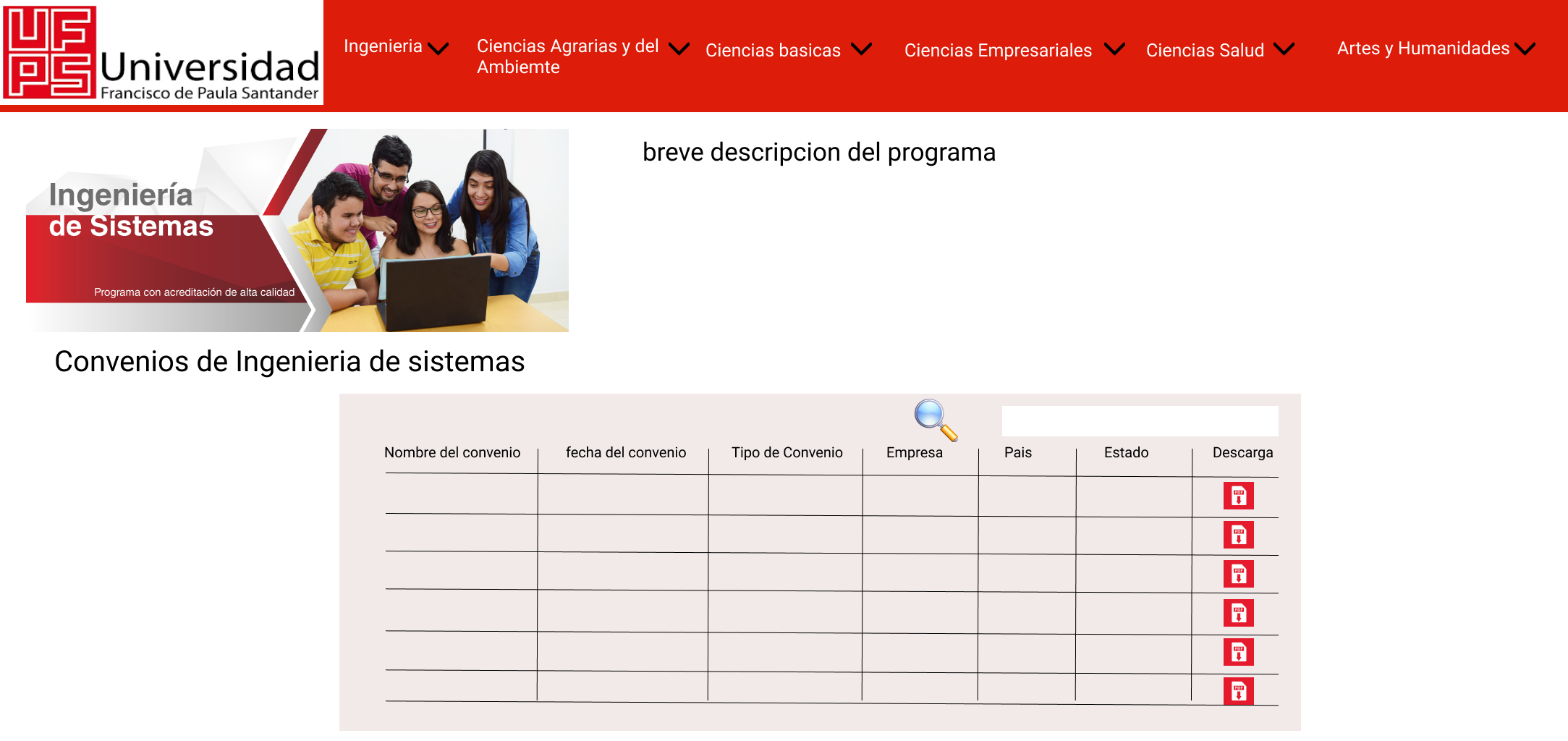
* 1. Comportamiento del Sistema

1. Diseño de Interfaces









* 1. Modelo de Contenido / Contexto de Interacción

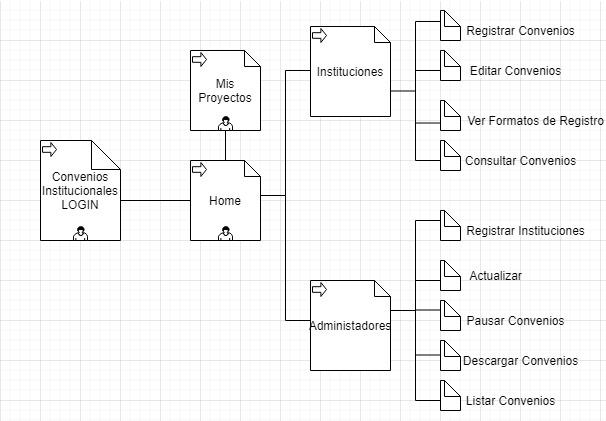
*[A continuación enliste los contextos de interacción. Cada contexto de interacción es un "lugar" donde los usuarios ven información y donde seleccionan comandos u opciones. En una interfaz gráfica del usuario, un contexto de interacción será eventualmente implementado como una ventana o un cuadro de diálogo. En otras aplicaciones, un contexto de interacción puede ser implementado como, por ejemplo, una página web, un menú de voz o un panel de control físico.*

*SUGERENCIA: Cada contexto de interacción es un modo exclusivo: el usuario puede usar solo un contexto de interacción a la vez. Todos los componentes dentro de un contexto son visibles y utilizables al mismo tiempo. Por ejemplo, si una ventana tiene tres pestañas, esos son los tres contextos de interacción porque solo una pestaña puede ser usada a la vez. ]*

*Para cada contexto de interacción enumere los componentes abstractos dentro de ese contexto. Cada componente es una pieza de información, o una facilidad de la interfaz del usuario. En una GUI, cada componente abstracto eventualmente se convertirá en una característica particular, pero la selección de características específicas viene después.*

*SUGERENCIA: La mayoría de los casos de uso de alta frecuencia deberían ser llevados a cabo solo en un contexto de interacción. Un caso de uso que requiera tres contextos de interacción puede ser difícil de utilizar. De cualquier forma, los contextos de interacción con demasiados componentes pueden ser difíciles de utilizar también.*

* 1. Mapa de Navegación



* 1. Consideraciones Técnicas / Contextualización Operacional

Conteste las siguientes preguntas

* ¿Cuáles son sus supuestos acerca de los dispositivos de salida?
* En los dispositivos de salida como pantalla e impresora, se podrá imprimir cualquiera de los archivos que se descarguen de la aplicación.
* ¿Cuáles son sus supuestos sobre los dispositivos de entrada que se usarán?
* Los dispositivos de almacenamiento como USB, Disco Duro Portable u otro dispositivo que cuente con el espacio suficiente para alojar el ejecutable, base de datos y documentación del Software PRACTIC, se podrá utilizar como dispositivo de respaldo de datos.
* ¿Cuáles son sus supuestos sobre la cantidad de tiempo que los usuarios invertirán en sus tareas?
* Después de un límite de tiempo máximo de 15 minutos el sistema sacarra cerrando sesión.
* ¿Qué administradores de ventanas, librerías de UI u otras tecnologías de UI utilizará?
* A continuación se describen las tecnologías de diseño que se utilizaron en el Sistema Web de Prácticas:

**BOOTSTRAP:** es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.

Bootstrap es el segundo proyecto más destacado en GitHub1​ y es usado por la NASA y la MSNBC entre otras organizaciones.

**HTML:**

Siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (). El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado World Wide Web Consortium, más conocido como W3C. Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo. El propio W3C define el lenguaje HTML como "un lenguaje reconocido universalmente y que permite publicar información de forma global". Por convención, los archivos de formato HTML usan la extensión .htm o .html.

**CSS:**

Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C es el encargado de formular la especificación de las hojas de estilo que servirá de estándar para los agentes de usuario o navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML. En este último podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "style". Las ventajas de utilizar CSS (u otro lenguaje de estilo) son: − Control centralizado de la presentación de un sitio web completo, con lo que se agiliza de forma considerable la actualización del mismo. − Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web remoto, con lo que aumenta considerablemente la accesibilidad. 29 − Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre. − El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

1. **Diseño de la base de Datos**
   1. **Modelo de datos**

* *Los estudiantes pueden repetir un curso (y obtener así dos registros para el mismo curso en su boleta) si y solo si tienen una calificación de "6.5" o menor, o el número del curso es 198, 199, 298 o 299.*
* *Una calificación mayor de "10" es válida solo para entradas de boletas durante o después de otoño de 1990. Antes de esta fecha, la calificación máxima posible es de "10".*
* *CONSIDERACIÓN LOGICA QUE NO PUEDE SER EXPRESADA EN EL DIAGRAMA*
* *CONSIDERACIÓN LOGICA QUE NO PUEDE SER EXPRESADA EN EL DIAGRAMA* 
  1. Almacenamiento de archivos

*[En esta sección conteste las siguientes preguntas: ¿Que información necesita ser almacenada en archivos?, ¿Cuáles son las convenciones para la estructura de directorios y nombrado de archivos?, ¿Qué controles de acceso al sistema de archivos se usaran?]*

**6.3 Almacenamiento Distribuido**

**MYSQL:**

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo,1​2​ y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL AB fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de doble licenciamiento anteriormente mencionado. La base de datos se distribuye en varias versiones, una Community, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones Enterprise, para aquellas empresas que quieran incorporarlo en productos privativos. Las versiones Enterprise incluyen productos o servicios adicionales tales como herramientas de monitorización y asistencia técnica oficial. En 2009 se creó un fork denominado MariaDB por algunos desarrolladores (incluido algunos desarrolladores originales de MySQL) descontentos con el modelo de desarrollo y el hecho de que una misma empresa controle a la vez los productos MySQL y Oracle Database.3​

Está desarrollado en su mayor parte en ANSI C y C++.4​ Tradicionalmente se considera uno de los cuatro componentes de la pila de desarrollo LAMP y WAMP.

**XAMPP:**

es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. El nombre es en realidad un acrónimo: X (para cualquiera de los diferentes sistemas operativos), Apache, MariaDB/MySQL, PHP, Perl. A partir de la versión 5.6.15, XAMPP cambió la base de datos MySQL por MariaDB, un fork de MySQL con licencia GPL.

El programa se distribuye con la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. A esta fecha, XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y Mac OS X.

Oficialmente, los diseñadores de XAMPP fueron los "Baiker y Anthony Corporation", quienes sólo pretendían su uso como una herramienta de desarrollo, para permitir a los diseñadores de sitios webs y programadores testear su trabajo en sus propios ordenadores cuando no tienen ningún acceso a Internet. En la práctica, sin embargo, XAMPP se utiliza actualmente como servidor de sitios web, ya que, con algunas modificaciones, es generalmente lo suficientemente seguro para serlo. Con el paquete se incluye una herramienta especial para proteger fácilmente las partes más importantes en una página.

**APACHE TOMCAT**

Tomcat es un contenedor de servlets que se utiliza en la Referencia oficial de la implementación para Java Servlet y JavaServer Pages (JSP). Las especificaciones Java Servlet y JavaServer Pages son desarrolladas por Sun Microsystems cuyas especificaciones vienen dadas por la JCP (Java Community Process). Apache Tomcat es desarrollado en un entorno abierto y participatorio, bajo la licencia de Apache Software License.

Para simplificar, podríamos decir que Apache Tomcat (o Jakarta Tomcat) es un software desarrollado con Java (con lo cual puede funcionar en cualquier sistema operativo, con su máquina virtual java correspondiente) que sirve como servidor web con soporte de servlets y JSPs.

Tomcat es mantenido y desarrollado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software License. Las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las versiones más recientes son las 7.x, que implementan las especificaciones de Servlet 3.0 y de JSP 2.2. A partir de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina.

Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.