**DISEÑO DE SOFTWARE**

**Proyecto: Desarrollo del diseño de un prototipo para un sistema web que permita la gestión de los grupos y elementos culturales en la**

**UPTC seccional Sogamoso.**

**CONTENIDO**

[1.) INTRODUCCIÓN 6](#_Toc72011275)

[1.1) Definición del problema 7](#_Toc72011276)

[1.2) Formulación 7](#_Toc72011277)

[1.3) Justificación 7](#_Toc72011278)

[1.4) Antecedentes 7](#_Toc72011279)

[1.5) Objetivos 8](#_Toc72011280)

[1.5.1) Objetivo general 8](#_Toc72011281)

[1.5.2) Objetivos específicos 8](#_Toc72011282)

[2.) MARCO TEORICO 8](#_Toc72011283)

[2.1) Marco Referencial 8](#_Toc72011284)

[2.2) Marco Legal 9](#_Toc72011285)

[3.) METODOLOGIAS 9](#_Toc72011286)

[3.1 Metodología de la investigación 9](#_Toc72011287)

[3.2 Metodología de desarrollo: xp 10](#_Toc72011288)

[3.3) Sistema Actual 11](#_Toc72011289)

[3.4) Sistema Propuesto 12](#_Toc72011290)

[4.) REQUISITOS FUNCIONALES 13](#_Toc72011291)

[4.1) Requisitos Funcionales 13](#_Toc72011292)

[4.2) Requisitos No Funcionales 17](#_Toc72011293)

[4.3) Tarjetas CRC 20](#_Toc72011294)

[5. DISEÑO DE SOFTWARE 24](#_Toc72011295)

[5.1) Esquema implementable de la base de datos 24](#_Toc72011296)

[5.2) Diccionario de datos 24](#_Toc72011297)

[6. PROTOTIPOS DE SOFTWARE 29](#_Toc72011298)

[6.1) Prototipos de instructor 29](#_Toc72011299)

[6.2) Prototipos de administrador de prestamos 33](#_Toc72011300)

[7. UML2 36](#_Toc72011301)

[7.1) Clases 36](#_Toc72011302)

[7.2) Paquetes 38](#_Toc72011303)

[7.3) Secuencias 38](#_Toc72011304)

[7.4) Casos De Uso 40](#_Toc72011305)

[8) ARQUITECTURA DE SOFTWARE 41](#_Toc72011306)

[8.1) Fundamentos de arquitectura 42](#_Toc72011307)

[8.2) Consideraciones 43](#_Toc72011308)

[8.3) Patrones de arquitectura 47](#_Toc72011309)

[8.4) EventSourcing 49](#_Toc72011310)

[8.5) Microservicios 49](#_Toc72011311)

[8.6) Administración cliente servidor 49](#_Toc72011312)

[8.4) Consumo web 49](#_Toc72011313)

[9) Construcción 50](#_Toc72011314)

[9.1) Lenguaje de construcción 50](#_Toc72011315)

[9.2) Fundamentos de construcción 50](#_Toc72011316)

[9.2.1) Biblioteca de construcción 50](#_Toc72011317)

[9.2.2) Minimizar complejidad 50](#_Toc72011318)

[9.2.3) Anticipación de cambios 52](#_Toc72011319)

[9.3) Repositorios 55](#_Toc72011320)

[9.4) Gestión de la configuración 55](#_Toc72011321)

[9.4.1) Planificación de construcción 55](#_Toc72011322)

[9.4.2) Métricas de construcción 56](#_Toc72011323)

[9.5) Reutilización de código 57](#_Toc72011324)

[10) Pruebas de verificación 58](#_Toc72011325)

[10.1) Pruebas de base de datos 58](#_Toc72011326)

[10.2) Pruebas de utilidad 58](#_Toc72011327)

[10.3) Pruebas Estáticas y dinámicas 60](#_Toc72011328)

[10.4) Pruebas Modernas 60](#_Toc72011329)

[10.5) Repositorio de administración de la configuración 61](#_Toc72011330)

[10.6) Capas de procesos ACS 61](#_Toc72011331)

[10.7) Administración de cambios 61](#_Toc72011332)

[10.8) Pruebas y anexos 63](#_Toc72011333)

[11) Métricas de diseño para webapps 63](#_Toc72011334)

[12) Estándares 63](#_Toc72011335)

[13) A tener en cuenta 64](#_Toc72011336)

[14) Conclusiones 64](#_Toc72011337)

[15) Bibliografía 64](#_Toc72011338)

[16) Infografía 64](#_Toc72011339)

# INTRODUCCIÓN

En una organización es de vital importancia llevar un orden de las personas que hacen parte de ciertos grupos, además de ello es importante llevar una gestión de los elementos con los que se cuenta. Por ello que a lo largo de este documento se registrará el proceso de avance que llevará necesario para la creación de un sistema de gestión cultural universitario apoyados en los contenidos dados en cursos pasados como ingeniería de requisitos, ingeniería de software I y cursos actuales como ingeniera de software II, tales como diseño preliminar, diseño detallado, diagramas UML y BPMN, metodologías, marcos de trabajo, procesos de construcción y verificación del producto, junto con todos los demás temas inmersos en la asignatura; Para el caso busca iniciar en trabajar en una implementación de para dicho sistema de gestión cultural, por lo que en primer lugar, se hace necesario hacer una correspondiente revisión documental de cómo se lleva el proceso de gestión de grupos culturales dentro de la UPTC seccional Sogamoso. Además de ello hay que hacer una revisión de la disponibilidad tecnológica para ello, y la documentación generada en el curso anterior.

## Definición del problema

El problema que se decide abordar para este caso, viene de un proyecto realizado en una asignatura anterior, en la cual se evidencio que es viable la realización de un sistema de gestión para elementos y grupos culturales de la UPTC en la seccional Sogamoso.

## 1.2) Formulación

¿Cuáles son los elementos de diseño y construcción de software que permitan desarrollar un sistema de gestión de los elementos y grupos culturales con los cuales cuenta la universidad en la seccional Sogamoso?

## 1.3) Justificación

Se ha decido realizar este proyecto debido a que tanto la inscripción y en general la gestión de todo lo relacionado a los grupos y a el préstamo de los elementos culturales se realiza por medio de plantillas, logrando que estas tareas se vuelvan tediosas e inseguras tanto para quienes se encargan de las mismas como para los estudiantes que hacen uso de ellas. Es por eso, que se ve necesaria la implementación de un sistema que facilite todo este proceso, necesidad que se verá reflejada por medio de algunas encuestas presentadas en secciones posteriores del documento

## 1.4) Antecedentes

Desde la creación de los grupos culturales de la UPTC en la seccional Sogamoso se ha visto una participación activa por parte de los estudiantes, quienes se inscriben de forma manual a dichos cursos por medio de planillas. Esto ha permitido que su formación no sea simplemente académica, sino que además les brinda una posibilidad de esparcimiento y diversión que hace del campo universitario un lugar propicio para su crecimiento tanto profesional como laboral.

Por otro lado, debido a que no todos los estudiantes de la universidad cuentan con los materiales necesarios para la práctica de estas actividades, y existe una gran cantidad de estudiantes interesados en aprovechar su tiempo libre en el aprendizaje de elementos como instrumentos musicales, pinturas, entre otros. La UPTC cuenta con un sistema manual de préstamos, en donde se hace uso de planillas que permiten llevar el registro de los mismos.

## 1.5) Objetivos

Para la continuación de este proyecto se plantean unos objetivos un poco más ambiciosos que en su predecesor.

### 1.5.1) Objetivo general

Crear el prototipo funcional de un sistema capaz de realizar la gestión cultural para la UPTC seccional Sogamoso, incluyendo los grupos culturales y el préstamo de los elementos culturales con los que cuenta la Universidad.

### 1.5.2) Objetivos específicos

* Determinar los fundamentos de la arquitectura de software para el prototipo en base a la información recolectada previamente.
* Determinar el patrón de arquitectura más conveniente para la construcción del prototipo.
* Realizar la construcción de cada uno de los módulos teniendo en cuenta la metodología XP seleccionada con anterioridad.
* Implementar pruebas sobre el sistema que determinen el cumplimiento de los requisitos previstos.

# MARCO TEORICO

## 2.1) Marco Referencial

**Cultura**: Según la rae el término cultura hace referencia a todo el conjunto de costumbres, modos de vida, conocimientos y grado de desarrollo artístico, científico, e industrial que identifican a un grupo de personas o sociedad en una época determinada.

**Gestión:** A lo largo de este documento se manejará el término gestión para hacer referencia al manejo de los elementos y grupos culturales que existen en la seccional Sogamoso mas no se referirá a conseguir nuevos elementos o grupos.

**Módulo:** El término módulo, para efectos del proyecto, hace referencia a las partes en las que se va dividir el sistema dependiendo de las funcionalidades que van a manejar. En este caso se tendrá el módulo de Gestión de grupos y el módulo de Préstamos, los cuales contarán con sus respectivas características.

**CRUD:** Son las siglas representativas para Crear, Recuperar o Leer, Actualizar y Eliminar; las cuales vienen del inglés Create, Read, Update y Delete. De igual manera, para el proyecto definen funciones claras que se deben tener para el correcto funcionamiento del problema.

**Módulo de préstamo:** cuando se menciona este módulo se hace referencia a la parte del sistema que contendrá todo lo relacionado al prestamos de elementos culturales de la seccional.

**Módulo de gestión cultural**: cuando se menciona este módulo se hace referencia a la parte del sistema que contendrá todo lo relacionado a la gestión de grupos culturales de la seccional.

## 2.2) Marco Legal

Para aspectos visuales de la realización del prototipo se trabajará teniendo en consideración la resolución 29543 de 2016 que concede el registro de algunos elementos de la identidad grafica de la UPTC, también la resolución 4995 de 2016 por el cual se adopta el manual de identidad grafica de la UPTC y el manual de identidad grafica de la universidad en el cual se incluye el uso adecuado de los logos y sus dimensiones, gamas cromáticas de los colores institucionales y combinaciones de logos, tipografías y colores. (Colombia, 2020)

# METODOLOGIAS

## 3.1 Metodología de la investigación

Para efectos de este proyecto fue necesario realizar una investigación mixta, es decir de tipo cuantitativo y cualitativo. Para ello, en primer lugar, se realizó la aplicación de una encuesta dirigida a los estudiantes de la Universidad Pedagógica y Tecnológica de Colombia (UPTC) seccional Sogamoso, en donde se incluían preguntas que se enfocaban en determinar la necesidad de un sistema que permitiera realizar la gestión de los grupos culturales de la UPTC y de los préstamos de los elementos culturales de la misma; además de incluirse preguntas abiertas que permitieran determinar los principales requisitos funcionales y no funcionales del sistema a desarrollar. Así mismo, se realizó el reporte de las historias de usuario con las que se obtuvo la información necesaria para completar el análisis y especificación de dichos requisitos, los cuales se verán reflejados en las secciones posteriores del documento.

## 3.2 Metodología de desarrollo: xp

Con base en lo expuesto en la sección anterior procedemos a seleccionar una metodología rápida que nos permita obtener el resultado más optimo apoyados en la información y el equipo de trabajo con el que contamos. Para el caso decidimos seleccionar la metodología xp o extreme programming ya que con base en el tiempo y las tareas con las que contamos se adapta de una manera correcta. A continuación, se procede a explicar los aspectos más importantes que se requieren para trabajar con esta metodología como lo son los roles, la definición de las historias de usuario y el cronograma de iteraciones que permitirán cumplir con el desarrollo del proyecto y la entrega oportuna del mismo.

**3.2.1 ROLES**

Para el uso de XP se hace necesario definir en primera instancia una serie de roles, debido a que este es un ejercicio de tipo académico contamos con limitados recursos en lo que respecta a equipo de trabajo, es por ello que los participantes del grupo repetirán roles. La metodología define como tal 6 roles que abordan al cliente, los programadores, testers, trackers, coach y un Big Boss de las cuales todas tienen sus quehaceres, como ya se había mencionado, al ser un ejercicio académico los integrantes del grupo cumpliremos los roles de cliente, programadores, testers, trackers y Big Boss y para el caso del coach el docente hará de este rol, ya que para ello se pueden aprovechar los espacios académicos brindados.

**3.2.2 HISTORIAS DE USUARIO Y CRONOGRAMA DE ITERACIONES**

Para iniciar con los procesos relacionados a la metodología se empiezan a definir una serie de historias de usuario que se presentaran en la sección 4.1, estas historias de usuario deberían ser brindadas por parte del cliente, pero para el caso fueron definidas con base a una investigación previa. Una vez tuvimos las historias de usuario en nuestro poder se procedió en reunión del grupo de trabajo a agrupar las historias de usuario de manera que se pudiera llegar a generar una iteración de desarrollo, considerando que las historias de usuario agrupadas puedan, en conjunto lleguen a formar una parte funcional del sistema. Con base en ese preámbulo y considerando el tiempo que se tiene disponible para la realización del proyecto, en una reunión en la que estuvieron presentes la mayoría de los involucrados en el proceso de desarrollo y se definen de manera preliminar 3 iteraciones a las cuales se les asignan sus correspondientes historias de usuario, así:

1. Iteración 1: En esta se decide abordar la historia de usuario número 1 y numero 2.
2. Iteración 2: La iteración dos contiene las historias de usuario número 3, numero 4 y numero 5. 3. Iteración
3. Por último, la iteración numero 3 contiene las historias de usuario 6 y 7.

Con ello, se obtuvo el calendario de iteraciones para realizar la elaboración del proyecto y entrega del mismo de la siguiente manera:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Abril | | | | Mayo | | | |
| **Actividades** | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1. Procesos de iteración 1 |  |  |  |  |  |  |  |  |
| 1. Procesos de iteración 2 |  |  |  |  |  |  |  |  |
| 1. Procesos de iteración 3 |  |  |  |  |  |  |  |  |
| 1. Pruebas sobre todo el sistema |  |  |  |  |  |  |  |  |
| 1. Organización del documento a entregar |  |  |  |  |  |  |  |  |
| 1. Entrega final del proyecto |  |  |  |  |  |  |  |  |

Es importante resaltar que al estar usando la metodología de desarrollo xp estas decisiones se toman de manera preliminar, mas no quiere decir que esto no sea susceptible a cambios, también por esa razón se decide hacer todo este proceso de una manera “superficial”, es decir, que no se hace un trabajo en exceso meticuloso. Además, se debe tener en cuenta que junto con el proceso de desarrollo se irá realizando el documento de presentación final para la asignatura en la que se está trabajando.

## 3.3) Sistema Actual

A día de hoy, en lo que respecta a la seccional Sogamoso en la parte de préstamos de elementos culturales el procedimiento a seguir es el siguiente.

* acercarse a la oficina.
* preguntar la disponibilidad.
* en caso de que esté disponible, entregar el Carné estudiantil.
* el encargado guarda el carné y entrega el elemento.
* en caso de que no esté disponible se da la posibilidad de separar algunos elementos para cierto horario.

El sistema que se maneja es bastante simple, ya que hasta hace un semestre este control se llevaba en hojas escritas a mano, pero desde hace poco se instaló un equipo de cómputo en esta oficina para llevar esta información tal vez en hojas de cálculo. En el sistema que se maneja actualmente el encargado debe hacer la verificación de la disponibilidad de los elementos llevando las cuentas de los elementos existentes y los elementos que ya han sido prestados, una desventaja que se tiene actualmente con este sistema es que deja a los estudiantes sin su carné, y en caso que lo necesiten no tienen la posibilidad de usarlo hasta que el préstamo sea finalizado, además de que se indica que el préstamo tiene un límite de una hora, y en muchos casos ya sea porque el encargado no se encuentra en su oficina, o porque el estudiante tiene que ir a clase o a realizar otra actividad olvida no se puede finalizar el préstamo en el tiempo determinado.

Para hablar a la parte referente a los grupos culturales podemos mencionar que el proceso para poder inscribirse es el siguiente:

* identificar la disponibilidad de grupos en una cartelera en el edificio de artes
* verificar los horarios del grupo deseado
* si alguno de los horarios es útil para el estudiante, se procede a asistir a la primera sesión
* en esta, el instructor entregará una planilla en la cual los asistentes escribirán sus datos personales como su código, nombre etc…

Este sistema tiene como ventaja su simplicidad, pero eso mismo puede actuar como desventaja ya que en caso de que se pierda o se dañe esta planilla los registros del grupo se perderán. es importante mencionar que estos grupos semanalmente tiene una disponibilidad relativamente amplia de horarios para que más estudiantes puedan llegar a participar, es importante mencionarlo porque puede llegar a suceder que los estudiantes se inscriban en las primeras semanas pero dejen de asistir a las sesiones cuando la intensidad académica aumente considerablemente y además de ello los datos del semestre inmediatamente anterior son usados para determinar la disponibilidad de grupos del nuevo semestre.

Estas dos partes de la universidad funcionan con sistemas totalmente independientes uno del otro y diferentes, pero nosotros queremos crear un sistema que reúna estos dos y permita manejarlos de una manera más ordenada, tal como se describe en el siguiente ítem.

## 3.4) Sistema Propuesto

Para mejorar el sistema descrito anteriormente se propone el desarrollo de un prototipo web que permita realizar tanto el manejo de todo lo relacionado a los grupos de cultura como danza, dibujo, teatro, entre otros; como el control de los préstamos de elementos culturales con su respectivo inventario. Lo anterior, tomando de base los conocimientos adquiridos en la asignatura de Ingeniería de Software.

La idea principal es desarrollar el prototipo mediante el uso de módulos bajamente acoplados, que permitan manejar el sistema de manera fácil y lo hagan fácilmente escalable y mantenible de forma que en un futuro se pueda llegar a implementar en la seccional. En consecuencia, el sistema tendrá en primer lugar el módulo de inicio, en donde se va a permitir la autenticación o registro de los usuarios; una vez el usuario pueda acceder al sistema se presentarán dos módulos, uno para los préstamos y otro para la gestión de los grupos culturales.

Dentro del módulo de préstamos se presentarán 5 sub módulos, que tendrán la función de permitir realizar los registros de préstamos, la actualización o modificación de algún registro, la búsqueda de registros pasados, la eliminación del préstamo en el momento de la entrega y el manejo de los inventarios, en donde se encontrarán otros dos módulos para el registro de nuevos materiales y el registro de materiales perdidos o que ya no presten alguna funcionalidad.

En lo que respecta al módulo de grupos, se encontrará compuesto por otros 5 sub módulos encargados de las funciones de creación de grupos, la actualización o modificación de los grupos registrados y las búsqueda y eliminación de los mismos, además se presentará el módulo encargado del registro de los participantes del grupo, en donde se tendrá en cuenta el número mínimo para que el grupo no sea eliminado. En caso de que dicho grupo no cuente con el número mínimo de estudiantes será eliminado de forma automática del sistema.

Además, se busca presentarle a los usuarios un sistema sencillo de manejar, con una interfaz amigable que permita el desarrollo de todas las funciones que se necesitan. Para ello se harán uso de formularios con los campos que se requieran para el ingreso de los datos que se van a almacenar; y botones para el manejo de eventos como el ingresar, en caso de que se esté accediendo al sistema, o el aceptar en el momento de completar el envió de los formularios. Se debe tener en cuenta, que para el ingreso al módulo o al módulo de grupos, se debe escoger en el módulo de inicio a la hora de ingresar, si quiere ingresar a préstamos a grupos, y el sistema validará la información y determinará si permite o no el acceso solicitado.

# REQUISITOS FUNCIONALES

## 4.1) Requisitos Funcionales

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RF01 |
| **Nombre del requerimiento** | Autenticación de usuario |
| **Características** | Los usuarios deben realizar su respectivo log in para ingresar al sistema. |
| **Descripción del requerimiento** | Los usuarios deben tener la opción de ingresar para poder acceder al sistema brindando datos como el correo electrónico institucional y su contraseña. |
| **Requisito NO funcional** | RNF01  RNF03  RNF04 |
| **Prioridad del requerimiento:** Alta | |
|  | |
| **Identificación del requerimiento** | RF02 |
| **Nombre del requerimiento** | Registro de usuario |
| **Características** | Los usuarios deben registrarse para ingresar al sistema. |
| **Descripción del requerimiento** | Los usuarios deben tener la opción de registro, en caso de que no se encuentren registrados, para poder acceder al sistema brindando datos como el correo electrónico institucional y su contraseña, nombre, cargo, etc. |
| **Requisito NO funcional** | RNF1  RNF02  RNF04 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RF03 |
| **Nombre del requerimiento** | Gestión de inventarios (CRUD) |
| **Características** | El usuario debe gestionar el inventario de instrumentos. |
| **Descripción del requerimiento** | El encargado de realizar los préstamos debe tener la opción de gestionar (Crear, recuperar, actualizar y eliminar) el inventario de los instrumentos culturales con los que cuenta la seccional para prestar a los estudiantes. |
| **Requisito NO funcional** | RNF01  RNF03 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RF04 |
| **Nombre del requerimiento** | Solicitud de préstamo (CRUD) |
| **Características** | EL usuario puede gestionar los préstamos. |
| **Descripción del requerimiento** | El encargado de realizar los préstamos debe tener la opción de realizar crear, eliminar, actualizar y buscar préstamos con campos que permitan ingresar la información que se pide al estudiante. Además, se debe validar la información solicitada y verificar si el préstamo es posible. |
| **Requisito NO funcional** | RNF01  RNF02  RNF03  RNF04 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RF05 |
| **Nombre del requerimiento** | Gestión de grupos culturales (CRUD) |
| **Características** | El usuario debe gestionar el inventario los grupos culturales de la seccional. |
| **Descripción del requerimiento** | El sistema debe brindar a los instructores la capacidad de gestionar sus grupos, desde crearlos, modificar horarios, añadir personas, eliminarlos etc. |
| **Requisito NO funcional** | RNF01  RNF02  RNF03  RNF04  RNF05 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RF06 |
| **Nombre del requerimiento** | Asignación de funcionalidades |
| **Características** | El usuario debe reportar perdidas. |
| **Descripción del requerimiento** | En caso de que alguno de los elementos que se prestan lleguen a ser perdidos o simplemente no se devuelvan el encargado de ese préstamo deberá tener la opción de buscar el préstamo realizado para seguir el protocolo asignado, además debe poder eliminar el elemento perdido del inventario |
| **Requisito NO funcional** | RNF01  RNF04  RNF06 |
| **Prioridad del requerimiento:** Alta | |

## 4.2) Requisitos No Funcionales

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RNF01 |
| **Nombre del requerimiento** | Interfaz del sistema |
| **Características** | El usuario debe contar con una interfaz web sencilla y comprensible. |
| **Descripción del requerimiento** | La interfaz del usuario se debe visualizar por medio de un entorno web, en donde se hagan uso de botones que facilitan la navegación por los diferentes módulos pantallas que presenta el sistema, las cuales deben contener la información inicial de cada uno de ellos. |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RNF02 |
| **Nombre del requerimiento** | Validación de información |
| **Características** | El sistema debe validar los datos proporcionados. |
| **Descripción del requerimiento** | El sistema debe validar la información registrada en cada uno de los campos de las gestiones. Teniendo en cuenta los campos que sean obligatorios o los tipos de datos que se deban ingresar en estos. En caso de haber un error en ello, el sistema debe notificarle al usuario por medio de un mensaje y permitirle realizar la corrección pertinente. |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RNF03 |
| **Nombre del requerimiento** | Desempeño |
| **Características** | El sistema debe garantizar un buen desempeño respecto al almacenamiento de datos. |
| **Descripción del requerimiento** | El sistema debe garantizar la confiabilidad en el almacenamiento de los datos de registro de usuarios, así como el resto de información que requiere ser almacenada. |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RNF04 |
| **Nombre del requerimiento** | Cargos de usuario |
| **Características** | El sistema debe permitir el acceso dependiendo del cargo que este desempeña |
| **Descripción del requerimiento** | El sistema debe garantizar que tanto los instructores como los encargados de los prestamos tengan acceso al módulo correspondiente a cada función. |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RNF05 |
| **Nombre del requerimiento** | Funcionalidad continua del sistema |
| **Características** | El sistema debe funcionar en todo momento |
| **Descripción del requerimiento** | El sistema debe estar en funcionamiento las 24 horas del día, los 7 días de la semana, para que el usuario pueda acceder a el en cualquier momento si lo considera pertinente. |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RNF06 |
| **Nombre del requerimiento** | Seguridad de la información |
| **Características** | El sistema debe garantizar la seguridad de toda la información que se proporcione. |
| **Descripción del requerimiento** | El sistema debe garantizar la seguridad de información como inventarios, datos personales y demás. |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento** | RNF07 |
| **Nombre del requerimiento** | Reglas de negocio |
| **Características** | El sistema debe cumplir con ciertas reglas proporcionadas por la universidad. |
| **Descripción del requerimiento** | 1. En caso de que uno de los grupos culturales no cuente con la inscripción mínima de personas o no se le haya asignado un profesor luego del tiempo límite, el sistema deberá eliminar este grupo por defecto, permitiendo de esta manera que solo se muestran los grupos activos. 2. El sistema debe borrar los registros de préstamos y grupos realizados al final de cada semestre de manera que el espacio no se malgaste y haga la experiencia de uso más cómoda para el usuario. |
| **Prioridad del requerimiento:** Alta | |

## 4.3) Tarjetas CRC

|  |  |
| --- | --- |
| **CRC** | |
| Número: 1 | Nombre: Autenticación |
| Usuario: Cliente | |
| Modificación de historia número: No aplica | Iteración asignada: 1 |
| Prioridad en negocio: alta | Riesgo de desarrollo: Bajo |
| Descripción: El ingreso a las plataformas de la universidad por lo general se realizan por medio del correo institucional asignado a cada uno de los funcionarios y la contraseña respectiva. | |
| Observaciones: | |

|  |  |
| --- | --- |
| **CRC** | |
| Número: 2 | Nombre: Autenticación |
| Usuario: Cliente | |
| Modificación de historia número: No aplica | Iteración asignada: 1 |
| Prioridad en negocio: alta | Riesgo de desarrollo: Bajo |
| Descripción: Todas las plataformas de la universidad manejan la misma plantilla de fondo, teniendo los mismo colores, logos y lemas característicos de la universidad. | |
| Observaciones: | |

|  |  |
| --- | --- |
| **CRC** | |
| Número: 3 | Nombre: Préstamo |
| Usuario: Cliente | |
| Modificación de historia número: No aplica | Iteración asignada: 2 |
| Prioridad en negocio: alta | Riesgo de desarrollo: Bajo |
| Descripción: Por lo general los encargados de los prestamos realizan el control de los mismo por medio de plantillas en donde registran el nombre de cada uno de los estudiantes que se dirigen a ellos para pedir prestado algún elemento. Dicha plantilla cuenta con campos para el nombre, código del estudiante, además tiene un espacio asignado para el código del funcionario que realizó el préstamo y otro espacio para el instrumento que fue prestado, junto con los campos respectivos de fecha y hora del préstamo. | |
| Observaciones: | |

|  |  |
| --- | --- |
| **CRC** | |
| Número: 4 | Nombre: Préstamo |
| Usuario: Cliente | |
| Modificación de historia número: No aplica | Iteración asignada: 2 |
| Prioridad en negocio: alta | Riesgo de desarrollo: Bajo |
| Descripción: El sistema de platillas que se maneja actualmente, al llevarse de forma manual, no permite realizar modificaciones en caso de algún error en los registros, lo que significa un problema para los funcionarios. Además, es un sistema que, aunque pueda funcionar de alguna manera, no es muy seguro y cómodo de manejar. | |
| Observaciones: Se recomienda a los diseñadores tener en cuenta la inclusión de funciones de modificación y búsqueda de los registros, además de una función que permita eliminar o almacenar en otro lado los préstamos que ya no se encuentren vigentes. | |

|  |  |
| --- | --- |
| **CRC** | |
| Número: 5 | Nombre: Préstamo |
| Usuario: Cliente | |
| Modificación de historia número: No aplica | Iteración asignada: 2 |
| Prioridad en negocio: alta | Riesgo de desarrollo: Bajo |
| Descripción: Los administradores de los prestamos deben llevar un control sobre los elementos que están disponibles para el préstamo, su estado y la cantidad de los mismos, por lo que se lleva a cabo una especie de inventario. Además, se debe hacer un reporte de los elementos que se pierden y cargar la deuda al estudiante correspondiente. En caso de que un elemento se encuentre en mal estado se debe reportar y eliminar del inventario. | |
| Observaciones: | |

|  |  |
| --- | --- |
| **CRC** | |
| Número: 6 | Nombre: Grupos |
| Usuario: Cliente | |
| Modificación de historia número: No aplica | Iteración asignada: 3 |
| Prioridad en negocio: alta | Riesgo de desarrollo: Bajo |
| Descripción: El registro de estudiantes en cada uno de los grupos lo realiza el instructor encargado del grupo por medio de plantillas en donde se tienen espacios para el nombre del grupo y el código del instructor. Además, en la plantilla están los espacios para los datos del estudiante, tales como nombre, código, carrera, correo, entre otros. | |
| Observaciones: | |

|  |  |
| --- | --- |
| **CRC** | |
| Número: 7 | Nombre: Grupos |
| Usuario: Cliente | |
| Modificación de historia número: No aplica | Iteración asignada: 3 |
| Prioridad en negocio: alta | Riesgo de desarrollo: Bajo |
| Descripción: Cada uno de los grupos debe contar con un número mínimo de participantes para mantenerse funcionando, por lo que muchas veces, aunque se abren los grupos estos no permanecen y los estudiantes y demás funcionarios no tienen forma de saber qué grupos se encuentran realmente disponibles en el momento. | |
| Observaciones: Seria de gran ayuda que el sistema contara con un registro y búsqueda de grupos de manera que se pueda tener más claro que grupos están activos, quienes son los encargados, los horarios y los participantes del mismo. | |

# DISEÑO DE SOFTWARE

## 5.1) Esquema implementable de la base de datos

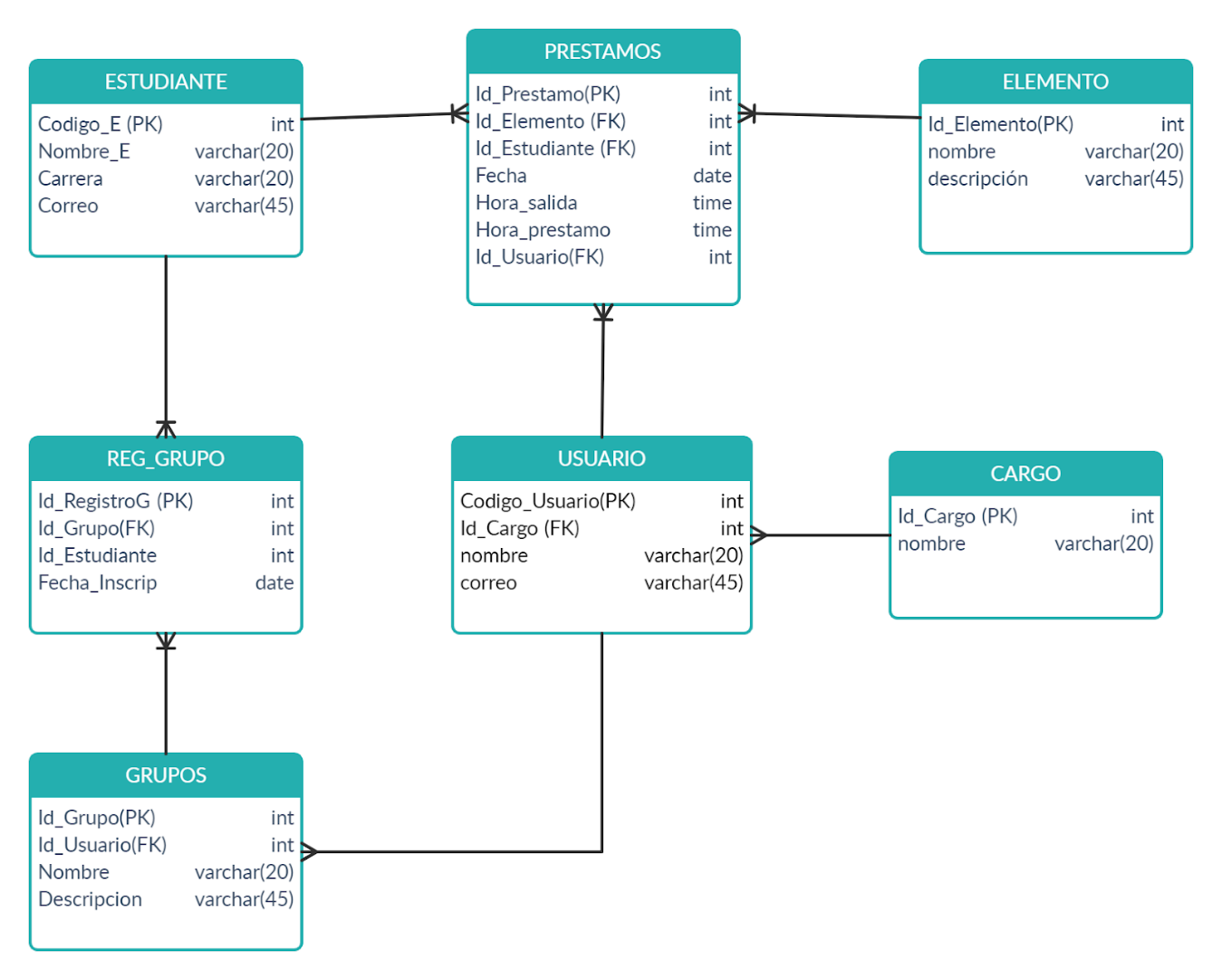
****

imagen 1. esquema de base de datos

En la imagen anterior se observa el modelo entidad relación usado para el manejo de datos del proyecto. Básicamente consta de una tabla con sus respectivos atributos para cada uno de los objetos que se crearán en la lógica del negocio y se añaden los registro y entidades transitorias para evitar que se presenten las relaciones muchos a muchos en el modelo. Esta será la estructura que se manejara en las dos bases de datos, debido a que en caso de los préstamos se almacenará en la base local en el momento en que se solicite el préstamo y será eliminada de allí y enviada a la base principal cuando se haga el retorno del elemento que se prestó. Ya para el caso de los grupos, estos se mantendrán almacenados en la base local durante el semestre académico y una vez este termine se limpiará dicha base almacenando el registro en la base de datos principal.

## 5.2) Diccionario de datos

**TABLA ESTUDIANTE:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descripción** | **obligación o condición** | **Tipo** | **Ejemplo** |
| **Codigo\_E** | Almacena el código de identificación de cada estudiante registrado. Debe ser único e irrepetible para cada uno de los registros. | Clave primaria no nula | int | 201711671 |
| **Nombre\_E** | Almacena el nombre de cada uno de los estudiantes registrados | No nulo | Varchar(45) | Erika Tinjaca |
| **Carrera** | Almacena el nombre de la carrera a la que pertenece el estudiante registrado | No nulo | Varchar(45) | Ingeniería en sistemas |
| **Correo** | Almacena el correo institucional del estudiante | No nulo | Varchar(45) | erika.tinjaca@uptc.ed  u.co |

**TABLA USUARIO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descripción** | **obligación o condición** | **Tipo** | **Ejemplo** |
| **Codigo:Usuario** | Almacena el código de identificación de cada uno de los usuarios directos del sistema (Instructores y encargados de préstamos). Debe ser único e irrepetible para cada uno de los registros. | Clave primaria no nula | int | Código de carnet del instructor de danzas |
| **Id\_Cargo** | Almacena el número de identificación del cargo asignado al usuario | Clave foránea No nula | int | 1001 |
| **Nombre** | Almacena el nombre del usuario registrado | No nulo | Varchar(20) | Nombre del  instructor del programa de danza |
| **Correo** | Almacena el correo institucional del usuario | No nulo | Varchar(45) | Correo del instructor del programa de danza |

**CARGO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descripción** | **obligación o condición** | **Tipo** | **Ejemplo** |
| **Id\_Cargo** | Almacena el código de identificación de cada uno de los cargos, los cuales pueden ser o instructor o prestamista | Clave primaria no nula | int | 1001 |
| **Nombre** | Almacena el nombre de del cargo | No nulo | Varchar (20) | Instructor |

**TABLA ELEMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descripción** | **obligación o condición** | **Tipo** | **Ejemplo** |
| **Id\_Elemento** | Almacena el código de identificación de cada uno de los elementos que se encuentran disponibles para préstamo | Clave primaria no nula | int | 101 |
| **Nombre** | Almacena el nombre del elemento registrado | No nulo | Varchar (20) | Guitarra |
| **Descripción** | Almacena la descripción detallada del elemento | No nulo | Varchar(20) | Guitarra eléctrica referencia 1345 |

**TABLA GRUPOS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descripción** | **obligación o condición** | **Tipo** | **Ejemplo** |
| **Id\_Grupo** | Almacena el código de identificación de cada uno de los grupos disponibles en la universidad | Clave primaria no nula | int | Código asignado al grupo de danzas |
| **Id\_Usuario** | Almacena el número de identificación del instructor encargado | Clave foránea No nula | int | código de usuario del instructor del grupo de danzas |
| **Nombre** | Almacena el nombre del Grupo registrado | No nulo | Varchar(20) | Grupo de danzas |
| **Descripción** | Almacena la descripción del grupo junto con los horarios que se manejan | No nulo | Varchar(45) | Grupo de danzas, lunes 8 a 10, miércoles 10 a 12 |

**TABLA REG\_GRUPOS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descripción** | **obligación o condición** | **Tipo** | **Ejemplo** |
| **Id\_Grupo** | Almacena el código de identificación del grupo al cual se está inscribiendo el estudiante | Clave Foránea no nula | int | Código asignado al grupo de danzas |
| **Id\_Registro** | Almacena el número de identificación del registro que se está realizando | Clave Primaria no nula | int | 01 |
| **Id\_Estudiantes** | Almacena el código del estudiante que está realizando la inscripción | Clave foránea no Nula | int | 201711674 |
| **Fecha\_ Inscripción** | Almacena la fecha en la que el estudiante realiza la inscripción | No nulo | Date | 24/09/2020 |

**TABLA PRESTAMOS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Descripción** | **obligación o condición** | **Tipo** | **Ejemplo** |
| **Id\_Préstamo** | Almacena el número de identificación del préstamo que se está realizando | Clave primaria no nula | int | 10 |
| **Id\_Usuario** | Almacena el número de identificación usuario que realiza el préstamo | Clave foránea no nula | int | Número de identificación del usuario que realiza el préstamo |
| **Id\_Estudiantes** | Almacena el código del estudiante que está realizando el préstamo | Clave foránea no Nula | int | 201711674 |
| **Id\_Elemento** | Almacena el código del elemento que se está prestando | No nulo | int | Código asignado a  la guitarra |
| **Fecha** | Almacena la fecha en la que se realiza el préstamo | No nulo | Date | 24/09/2020 |
| **Hora\_prestamo** | Almacena la hora en la que se realiza el préstamo | No nulo | Time | 4:36 p.m. |
| **Hora\_retorno** | Almacena la hora en la que el estudiante retorna el elemento | No Nulo | Time | 6:40 p.m. |

# PROTOTIPOS DE SOFTWARE

## 6.1) Prototipos de instructor



imagen 2. prototipo alta fidelidad 1

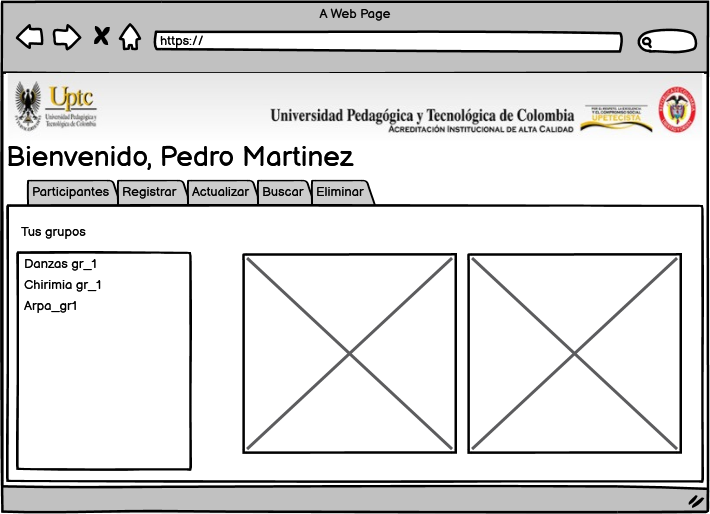


imagen 3. prototipo alta fidelidad 2

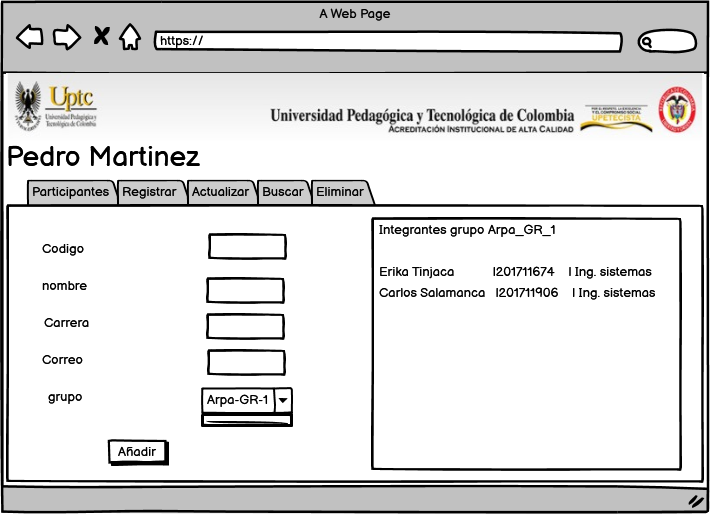


imagen 4. prototipo alta fidelidad 3

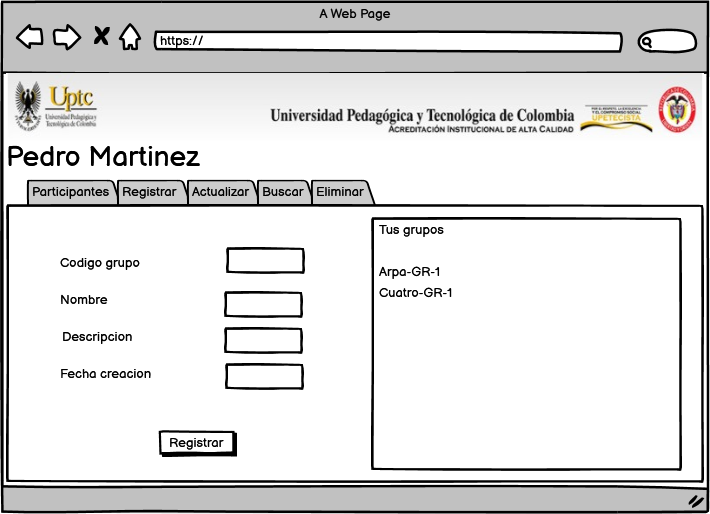


imagen 5. prototipo alta fidelidad 4

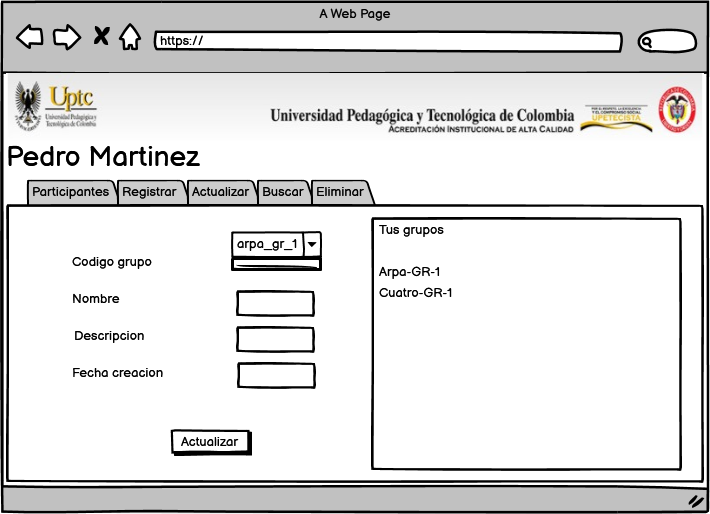


Imagen 6. prototipo alta fidelidad 5

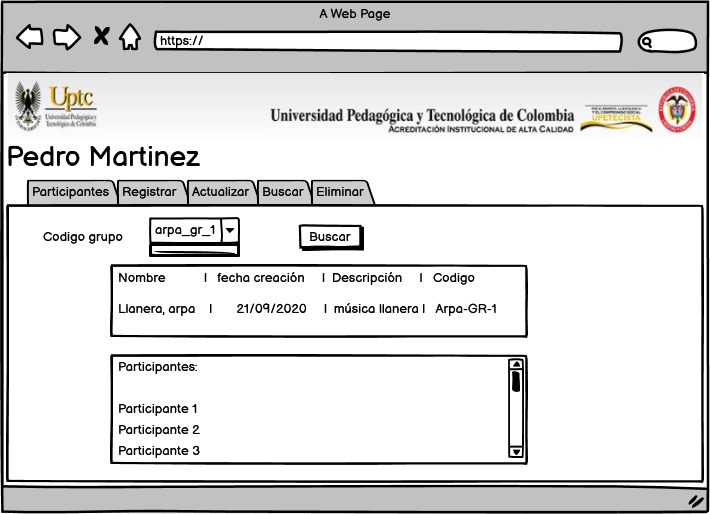


imagen 7. prototipo alta fidelidad 6

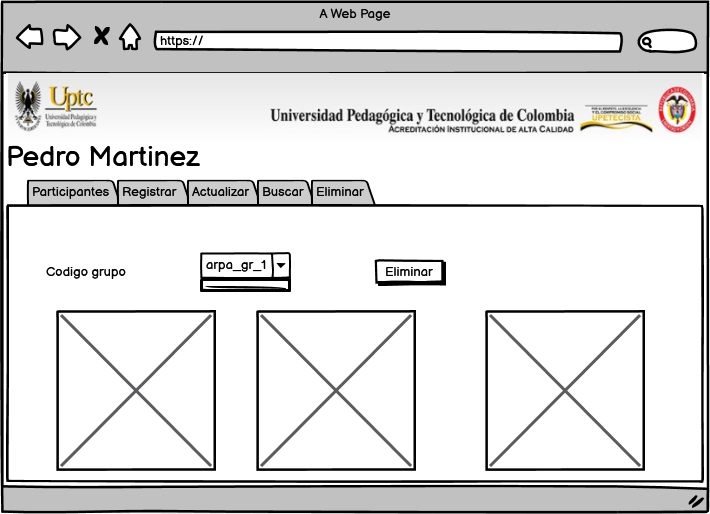


imagen 8. prototipo alta fidelidad 7

## 6.2) Prototipos de administrador de prestamos

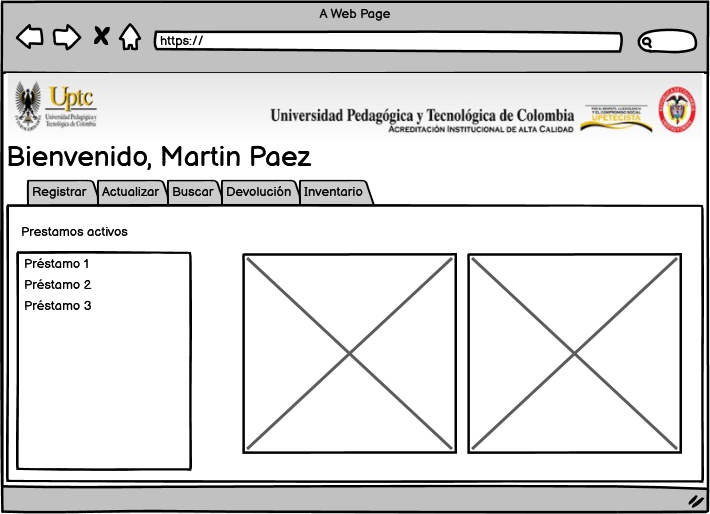


imagen 9. prototipo alta fidelidad 8

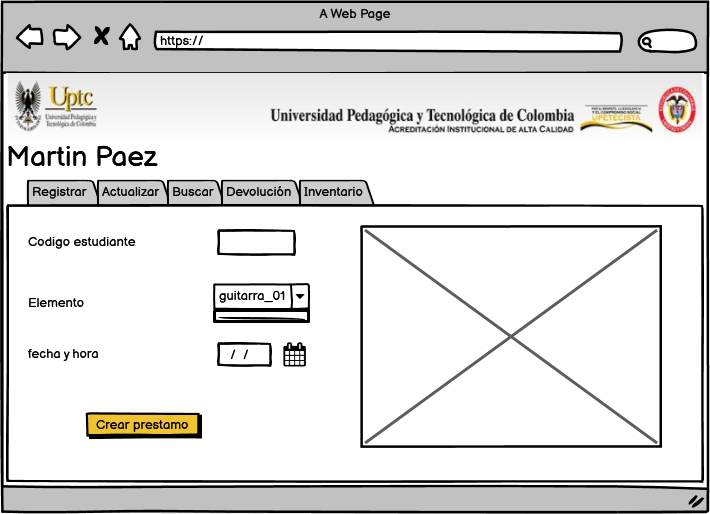


imagen 10. prototipo alta fidelidad 9

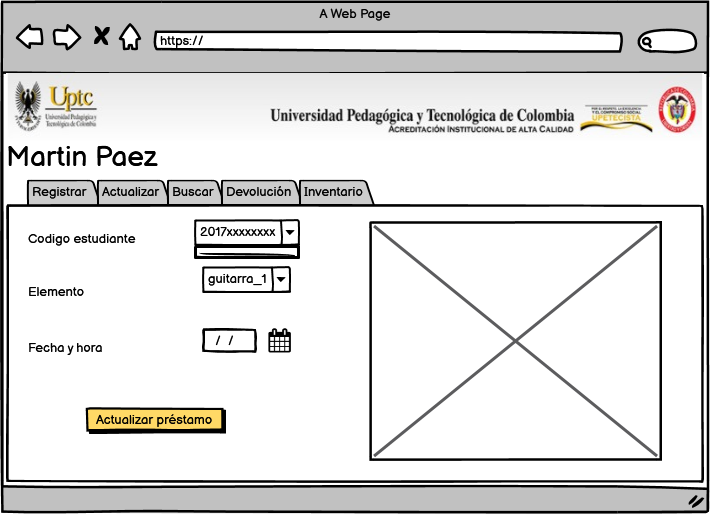


imagen 11. prototipo alta fidelidad 10

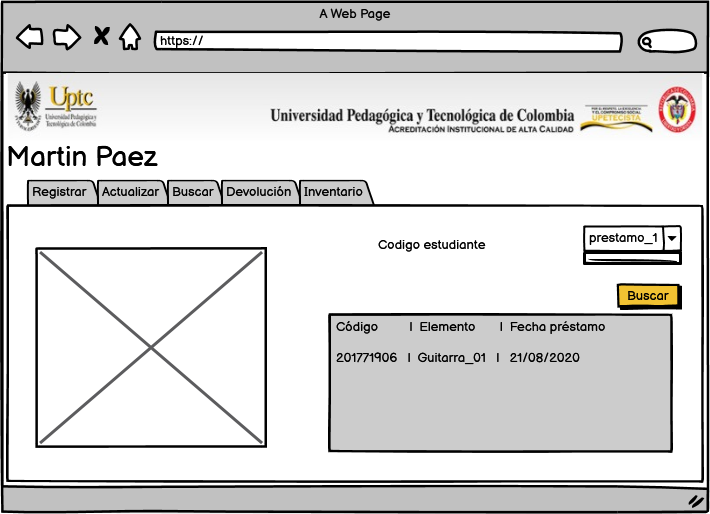


imagen 12. prototipo alta fidelidad 11

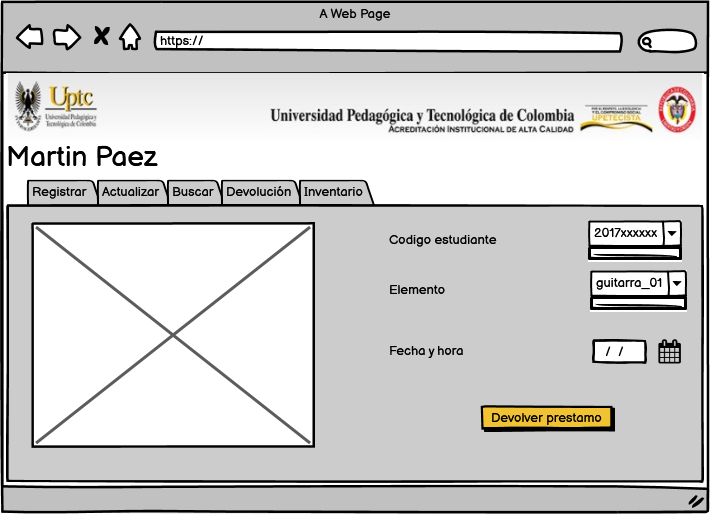


Imagen 13. prototipo alta fidelidad 12

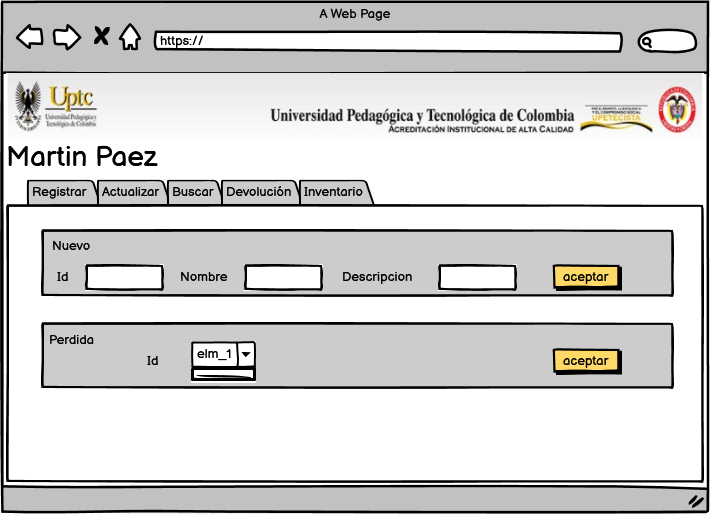


imagen 14. prototipo alta fidelidad 13

Los prototipos presentados desde la imagen 31 hasta la imagen 43 muestran la alta fidelidad, sin embargo, tampoco son un diseño exacto de la interfaz final, debido a impedimentos de la herramienta seleccionada para la generación de estos prototipos. Cabe mencionar que en estos modelos puede llegar a cambiar la identidad gráfica, ya que aún no se tiene diseñado un encabezado dirigido a este proyecto y se hizo uso del encabezado del ingreso al sistema SIRA de la universidad con el fin de contextualizar.

# UML2

Para el caso, al haber seleccionado la metodología xp, se solicita la realización de 4 tipos de diagramas UML2, los cuales son los diagramas de paquetes, casos de uso, secuencias y clases los cuales se mostraran a continuación.

## 7.1) Clases

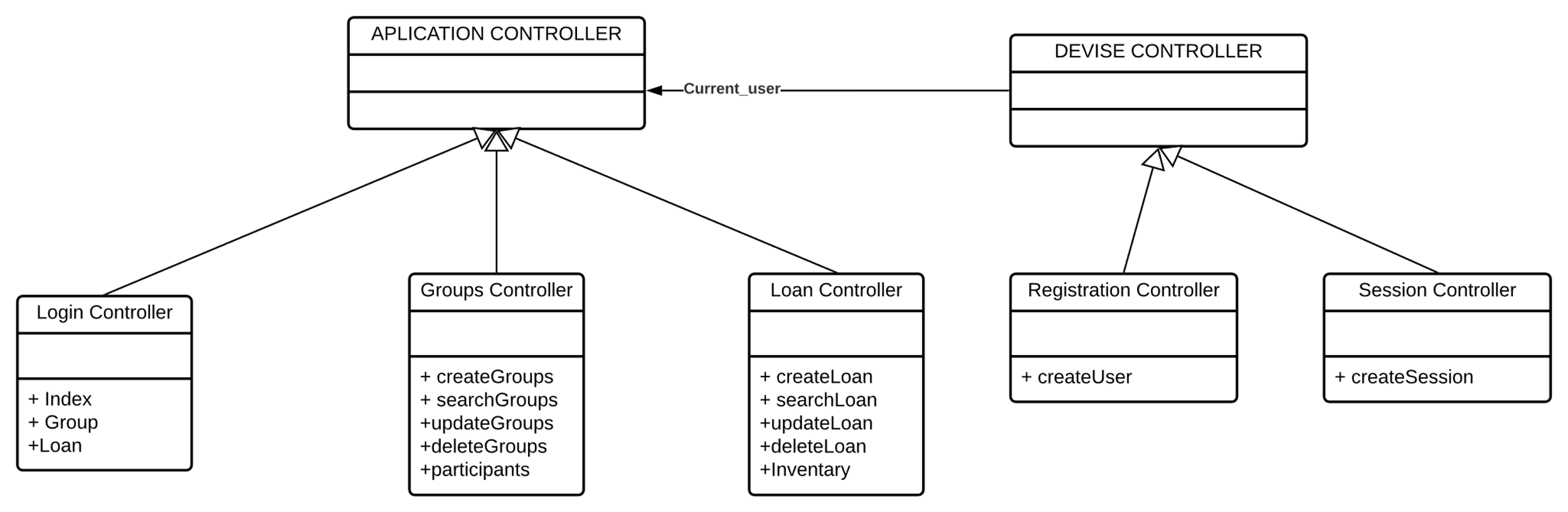


Imagen.15. Diagrama de clases del prototipo

En la Img. 6. Se presenta de manera detallada el diagrama de clases correspondiente a la lógica que se manejará para la realización del prototipo. Dentro de este se presentan las clases relacionadas a los diferentes objetos que se obtuvieron de la abstracción de datos y en donde se muestran los atributos necesarios para uno. Además, se observa el uso de dos clases grandes, proporcionadas por Ruby on Rails, dentro de las cuales se hace la gestión de grupos, prestamos, sesiones y registro de usuarios necesarios para la construcción del software deseado.

## 7.2) Paquetes

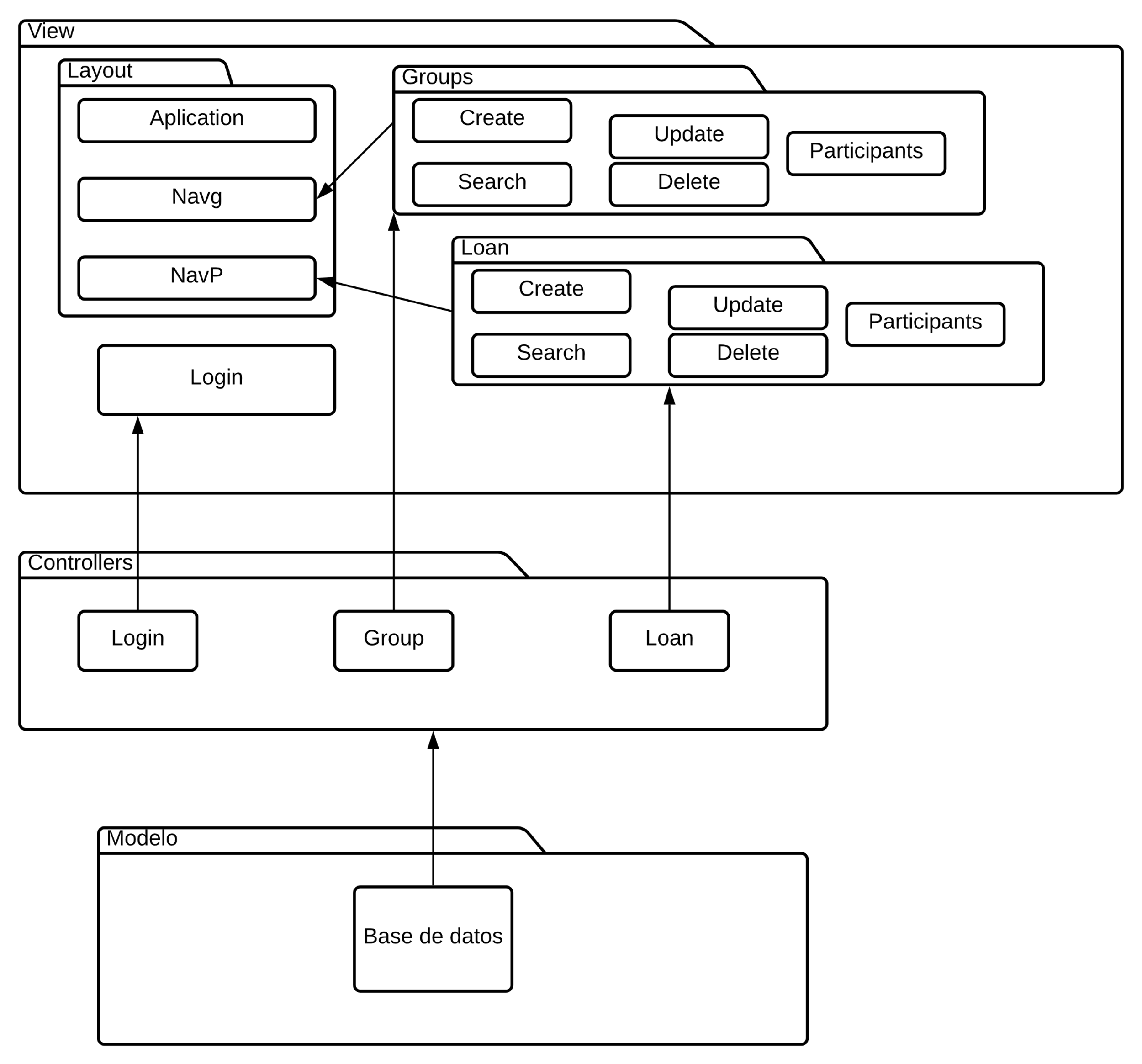


Imagen.16. Diagrama de Paquetes del prototipo

En la Img. 9. Se presenta de manera detallada el diagrama de paquetes, el cual nos especifica los tres paquetes centrales correspondientes a cada una de las capas de la arquitectura, y nos define los elementos u objetos, con sus respectivas conexiones, que se encuentran dentro de los diferentes paquetes, así como la manera en la que estos elementos se conectan con los elementos de los demás paquetes.

## 7.3) Secuencias

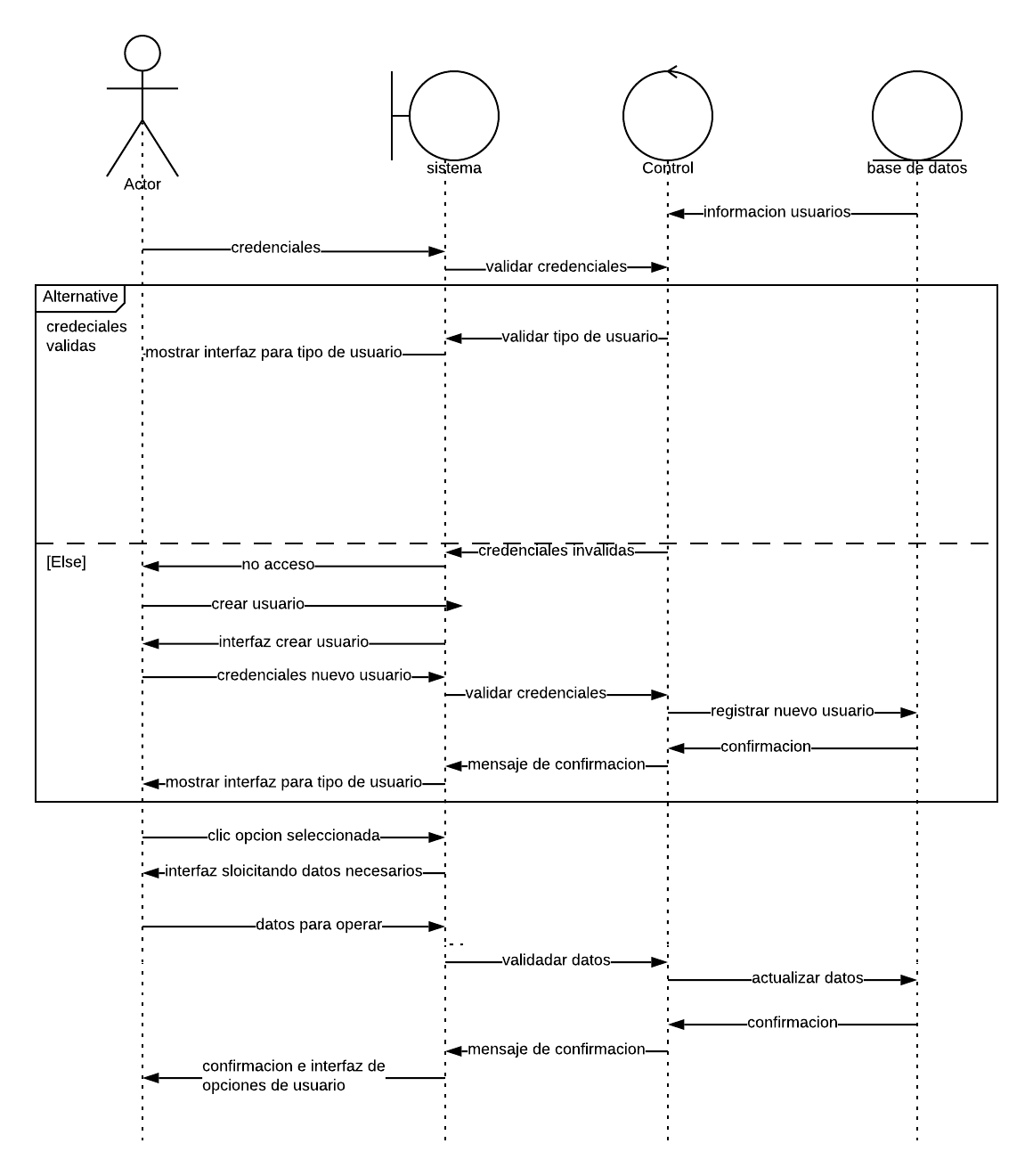


Imagen.17. diagrama de secuencias del prototipo

La imagen 12 refiere al diagrama de secuencias propuesto para el prototipo, este refleja cómo es la secuencia de comunicación para que el sistema llegue a funcionar, muestra como el usuario envía mensajes y hasta que elemento llegan, así como las respuestas de estos si es que las envían al usuario.

## 7.4) Casos De Uso

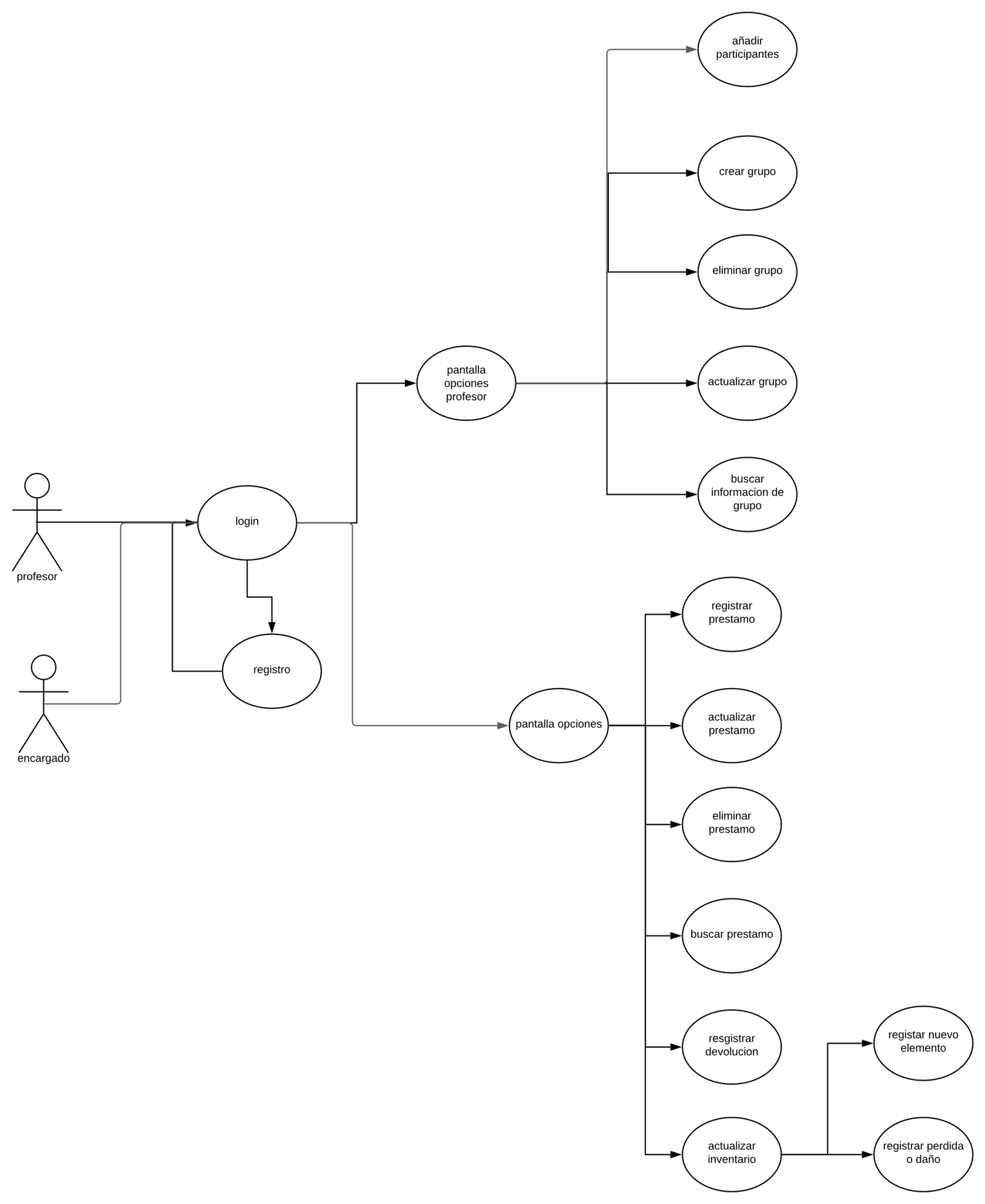


Imagen.18. diagrama de casos de uso del prototipo

El diagrama de casos de uso representa los posibles usos que puede llegar a darle cada uno de los posibles usuarios del sistema, refleja las posibles opciones que existen para cada tipo de usuario identificado para el sistema.

# 8) ARQUITECTURA DE SOFTWARE

Para efectos del proyecto se va a usar una arquitectura de tres capas, en donde la presentación se realizará por medio de una interfaz de Usuario realizada con HTML, Css y JavaScript. La capa de lógica de negocio, será realizada en lenguaje Java, haciendo uso de la Programación Orientada a objetos, usando clases para cada objetos y métodos para el desarrollo de las funcionalidades. Por último, para la persistencia se hará uso de una base de datos que nos permita el almacenamiento y la recuperación de datos para cubrir las necesidades del sistema.

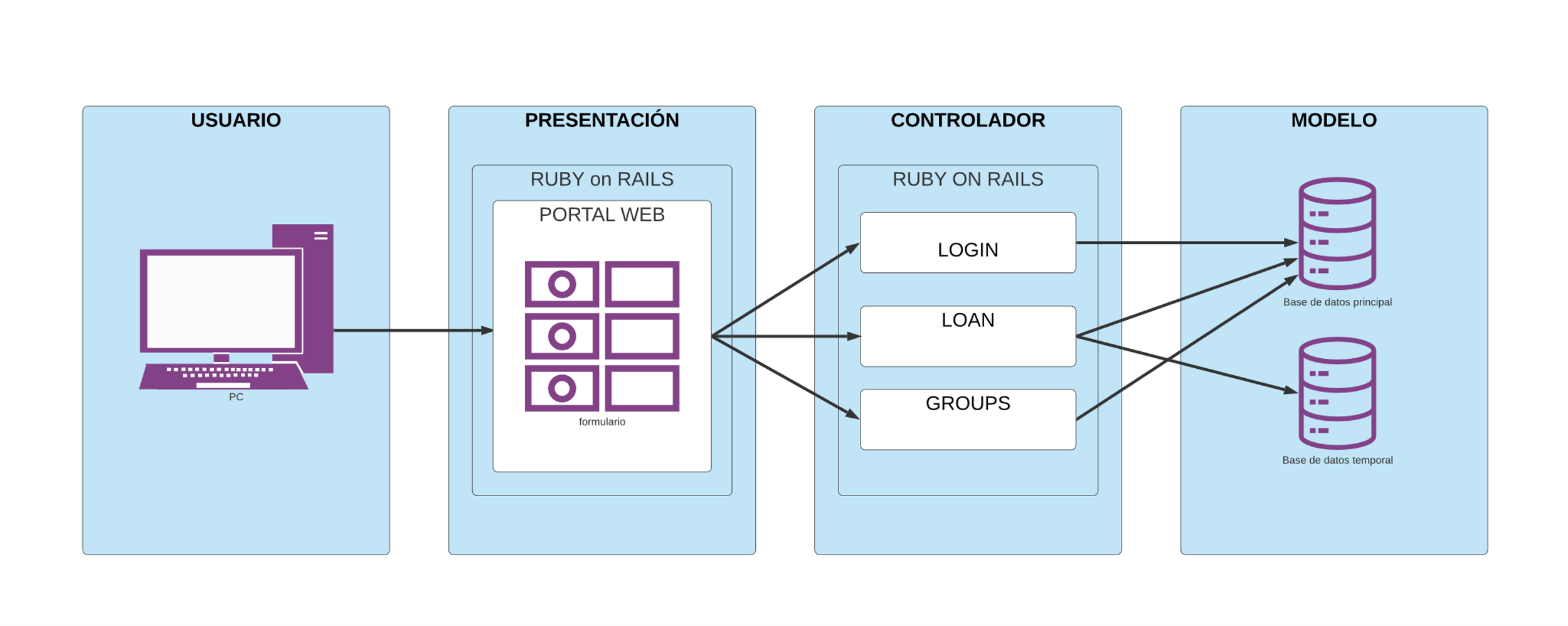


imagen 19. Estructura y arquitectura lógica del sistema

En la imagen 19 se presenta la estructura y arquitectura lógica del sistema, la cual, atendiendo a lo especificado por el modelo de tres capas, se presenta la parte del usuario el cual debe contar con el computador para acceder al sistema. Seguido de ello se presenta la capa de **Presentación**, dentro de la cual se manejará el portal web o la interfaz gráfica con la cual interactuará el usuario y que se realizará por medio de herramientas como HTML.ERB, CSS y las ayudas propias de RoR. Luego se muestra la capa lógica, la cual se encargará de manejar y ejecutar todas las funcionalidades del sistema y se pretende realizar usando Ruby on Rails, en especial lenguaje Ruby como el lenguaje de programación del sistema, allí se encontrarán los diferentes funciones, atributos y clases que nos permitan cumplir con las necesidades del cliente. Por último, se presenta la capa de modelo, que se compone por las dos bases de datos con las que contará el sistema.

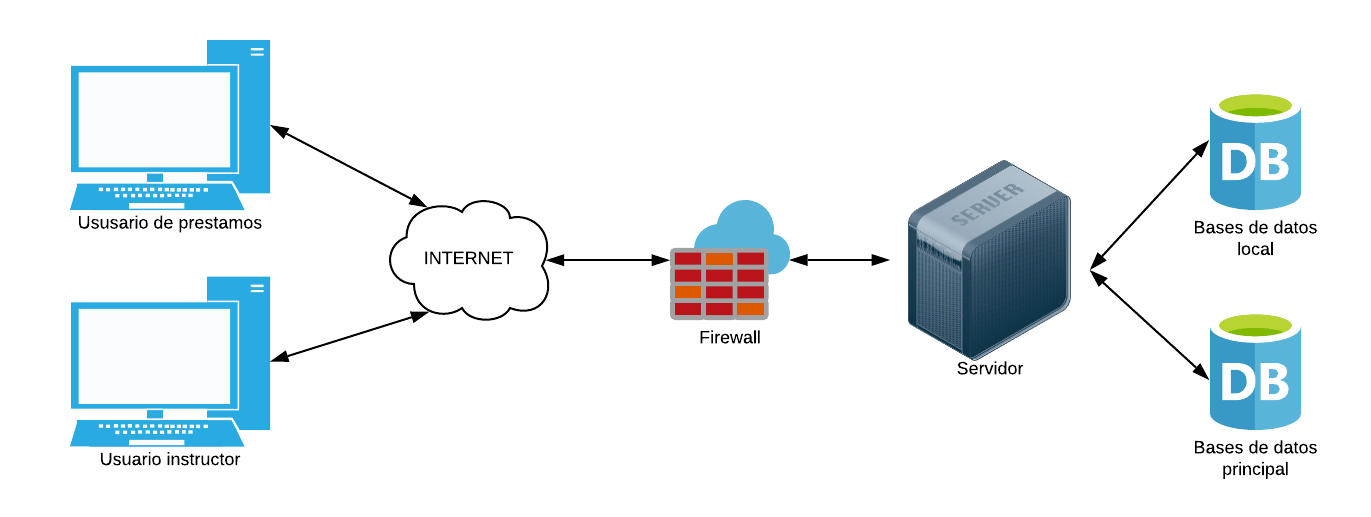


imagen 20. Estructura y arquitectura física del sistema

En la imagen 20. Se presenta la arquitectura física del sistema, la cual se inicia desde el equipo o los equipos de los usuarios quienes hacen la solicitud de ingreso, seguido de ello la información es enviada vía internet hacia el servidor, el cual se comunica con las dos bases de datos, enviando la información pertinente y recibiendo las repuestas esperadas para devolverlas y presentarlas al usuario.

## 8.1) Fundamentos de arquitectura

* **Idoneidad funcional**

La idoneidad funcional es un atributo que representa la capacidad que tiene un producto de software de brindar al usuario las funciones necesarias para satisfacer las necesidades del mismo. A continuación, se presentará una tabla en donde se presentan cada uno de las funcionalidades principales del sistema y seguido se presenta una breve descripción de cómo se dará cumplimiento a dicha funcionalidad.

|  |  |
| --- | --- |
| **Funcionalidad** | **Desarrollo** |
| **Ingreso de los usuarios** | El sistema contará con un módulo de login, el cual por medio de un usuario y una contraseña permitirá que tanto los usuarios encargados de los préstamos y los usuarios encargados de los grupos puedan acceder a las respectivas funcionalidades del mismo. Es importante aclarar que una vez el usuario realicé la autenticación respectiva el sistema se encargará de enviarlo al módulo respectivo. |
| **Gestión de Grupos (CRUD)** | Las funcionalidades respectivas a la gestión de los grupos, es decir Crear, Buscar, Actualizar y Eliminar; se garantizan por medio de un módulo de gestión de grupos, el cual contará con un submodulo dedicado a cada una de las funcionalidades antes mencionadas y en los cuales se encontrarán los campos y botones respectivos para facilitar el uso y navegabilidad de los usuarios. |
| **Gestión de Prestamos (CRUD)** | Las funcionalidades respectivas a la gestión de los préstamos, es decir Crear, Buscar, Actualizar y Eliminar; se garantizan por medio de un módulo de gestión de préstamos, el cual contará con un submodulo dedicado a cada una de las funcionalidades antes mencionadas y en los cuales se encontrarán los campos y botones respectivos para facilitar el uso y navegabilidad de los usuarios. |
| **Gestión de Inventario (CRUD)** | Las funcionalidades respectivas a la gestión de los inventarios, es decir Crear, Buscar, Actualizar y Eliminar; se garantizan por medio de un submodulo inventarios, disponible en el módulo de gestión de préstamos, el cual contará con los campos y botones necesarios que permitan desarrollar cada una de las funcionalidades referentes al CRUD. |

* **Compatibilidad**

La compatibilidad hace referencia a la capacidad que tienen los sistemas y sus componentes para intercambiar información y realizar sus funciones cuando comparten el mismo entorno de software o hardware con otros sistemas. Para el caso de este sistema, lo que busca es lograr tener un sistema interoperable, de manera que cada uno de los componentes del mismo puedan transportar la información de manera segura y eficaz; es por ello que las conexiones y llamados a los diferentes servicios, como por ejemplo a la base de datos) deben darse de manera transparente y evitando al máximo el riesgo de fallos, de esta forma se garantiza en buena parte la compatibilidad del sistema.

## 8.2) Consideraciones

Primero, para poder definir objetivamente un alcance debemos considerar que es un ejercicio académico y que como tal no se llegara a implementar un sistema en su totalidad, sino que se realizara una versión que permita navegar en las funcionalidades de una manera clara y aproximada a un sistema real que cumpla este propósito. Con real se hace referencia a un sistema oficial proporcionado por la universidad. Con eso estando claro podemos mencionar que el alcance de este proyecto será llegar hasta obtener una aplicación web que permita explorar las funcionalidades básicas que fueron reveladas en una investigación realizada previamente, pero sin llegar a ser una aplicación implementada en su totalidad. Lo importante realmente de este ejercicio académico más que la implementación de un sistema tangible, es poder hacer un acercamiento a una aplicación de todo el conocimiento teórico obtenido en esta serie de materias de la línea de ingeniería de software.

Dentro de las limitaciones del proyecto encontramos una de las más grandes la cual es el tiempo ya que juega en nuestra contra debido a que para poder llegar hacer un desarrollo de un sistema de este tipo y siendo la primera vez se realiza todo el proceso de ingeniería, y además aun estando estudiándolo el avance realmente no es tan rápido como las personas lo esperarían. Otro factor importante dentro de las limitaciones también podemos encontrar que todo el proceso tanto de desarrollo como de ingeniería este realizado solo por los dos integrantes del grupo de trabajo, esto genera que haya múltiples roles asignados a las mismas personas lo cual dentro del mundo real no es una practica recomendada. También se presentan limitaciones de tipo económico ya que para poder realizar la implementación de un sistema lo más próximo a ser real se debe hacer una inversión de capital, por ejemplo, para el alquiler de un servidor donde corra y responda las peticiones que le solicitemos, entonces lo que se hace o se tiende a hacer en un ámbito académico es usar la maquina de uno de los integrantes como servidor. Por ultimo se encuentran limitaciones de tipo personal las cuales no son tema a profundizar en el presente documento, pero influyen negativamente en el desarrollo de este proyecto.

Refiriéndonos a los StakeHolders es importante mencionar que el docente de la asignatura nos dio el privilegio de escoger la temática sobre la que queríamos trabajar, por esto podemos decir que los participantes del grupo somos stakeholders y podemos definir las tecnologías con las que nos es más cómodo trabajar, además de definir los atributos de calidad de la manera más pertinente para el proyecto. Con esto no se pretende dar a entender que el proyecto se vaya a desarrollar a la ligera, sino que en el tema de coordinación de los stakeholders no se presentaran mayores dificultades.

Se determina pertinente seguir con el proyecto con los resultados obtenidos hasta el momento, y esto lleva inevitablemente a iniciar a trabajar con el diseño y/o creación de la arquitectura con la que se trabajara hasta finalizar el proyecto. Se definen unas estrategias que se van a utilizar, donde se tuvieron en cuenta los puntos de vista de los tres interesados en el proyecto, mencionados previamente en el documento y se encuentran requisitos funcionales con más peso aquellos que hablan sobre la gestión tanto de grupos como de préstamos (RF4, RF5), así como el manejo del inventario de los elementos con los que cuenta la universidad (RF3). De allí se derivan las diferentes vistas para las cuales se tendrán presentes los siguientes atributos de calidad:

* + **Usabilidad:**

|  |  |
| --- | --- |
| **Atributo** | **Descripción** |
| **Reconocimiento de idoneidad** | El usuario identificará y podrá acceder a las funcionalidades a través de los diferentes módulos y sus respectivos sub módulos; permitiendo desempeñar sus tareas de manera más ágil y sencilla. El objetivo es generar un sistema que contenga en sus diferentes módulos los campos y botones con los nombres pertinentes para que el usuario pueda navegar de manera sencilla por la parte del sistema a la que podrá tener acceso. |
| **Curva de aprendizaje** | Este atributo se presentará en el sistema por medio de una interfaz gráfica intuitiva y amigable con el usuario; el cual por medio de botones de navegabilidad y con mensajes que le permitan identificar cuando falla y por qué. |
| **Operabilidad** | La operabilidad del sistema será bastante sencilla para el usuario. En los campos respectivos el usuario deberá ingresar los datos pertinentes; y por medio de los botones y la barra de navegación este podrá acceder a todos las funcionalidades disponibles para él. |
| **Protección de errores** | Para la protección de errores el sistema contará con la validación respectiva para cada uno datos que el usuario requiera enviar en cada funcionalidad; en caso de presentarse algún error el sistema enviará un mensaje al usuario de manera que este identifique y corrija su error. |
| **Estética de la interfaz** | La estética de la interfaz debe ser acorde a las interfaces presentadas por la universidad, debido a que es un sistema pensado para esta organización. |
| **Accesibilidad** |  |

* + **Confiabilidad:**

|  |  |
| --- | --- |
| **Atributo** | **Descripción** |
| **Madurez** | La madurez del sistema se ve reflejada en el funcionamiento del sistema y todas sus funciones cuando este no presenta fallos o situaciones inesperadas. En esos momentos el sistema permitirá las gestiones del CRUD necesarias en cada uno de los modulos, así como las funciones de autenticación y registro, sin inconveniente alguno y notificando al usuario si existe alguna inconsistencia de los datos que registra. |
| **Disponibilidad** | Al ser un prototipo para una aplicación web, el sistema se encontró disponible al usuario desde cualquier dispositivo en cualquier momento; siempre y cuando el usuario cuente con conexión a internet. |
| **Tolerancia a fallos** |  |
| **Capacidad de recuperación** |  |

* + **Seguridad:**

|  |  |
| --- | --- |
| **Atributo** | **Descripción** |
| **Confidencialidad** | El sistema garantizará al usuario que el trato de sus datos se hará acorde a las políticas de seguridad brindadas por todos los sistemas de la Universidad. |
| **Integridad** | La integridad de los datos se manejará manejando una buena comunicación entre las partes o capas del sistema de manera que los datos dispuestos en el sistema se mantengan completos y sin alteraciones de ningún tipo. |
| **Comprobación de hecho** | Este atributo se manejará por medio de la validación de los campos, de manera que en el momento en que el usuario los envié y se registren en el sistema, estos sean coherentes y acorde a la información que se está almacenando. |
| **Traza de responsabilidad** |  |
| **Autenticidad** | El sistema debe garantizar que no cualquier persona pueda acceder y llegué a acceder a los datos que allí se encuentran; por lo que solo aquellos usuarios que se encuentren registrados pueden acceder al módulo respectivo y deberán realizar el proceso de autenticación para acceder a las funcionalidades respectivas. |

* + **Mantenibilidad**

|  |  |
| --- | --- |
| **Atributo** | **Descripción** |
| **Modularidad** | El sistema propuesto cuenta con tres módulos principales. El primero correspondiente al login de la aplicación; el segundo destinado a las funcionalidades gestión de préstamos e inventarios; y el tercero diseñado para la gestión de los grupos culturales. A su vez, dichos módulos contarán con un sub modulo para cada una de las funciones de CRUD en lo que respecta a prestamos y a grupos; de manera que se maneje un bajo acoplamiento y una alta cohesión en todos los módulos existentes. |
| **Reusabilidad** | Debido a el código fuente de los diferentes módulos puede llegar a verse de forma general como un código útil para la gestión de cualquier producto; su reusabilidad resulta sencilla no solo para los demás módulos del sistema sino también para su uso en otros sistemas con una funcionalidad similar. |
| **Analizabilidad** |  |
| **Modificabilidad** | Los que se busca con este sistema es que este pueda llegar a modificarse de manera sencilla y siempre en busca de la mejora o escalabilidad, sin necesidad de que los cambios realizados en un módulo dañen los demás. |
| **Testabilidad** | En cada una de las iteraciones que se llevaran a cabo se realizarán las pruebas respectivas, de manera que se lleguen a probar cada uno de los módulos y funcionalidades del sistema. |

* + **Portabilidad**

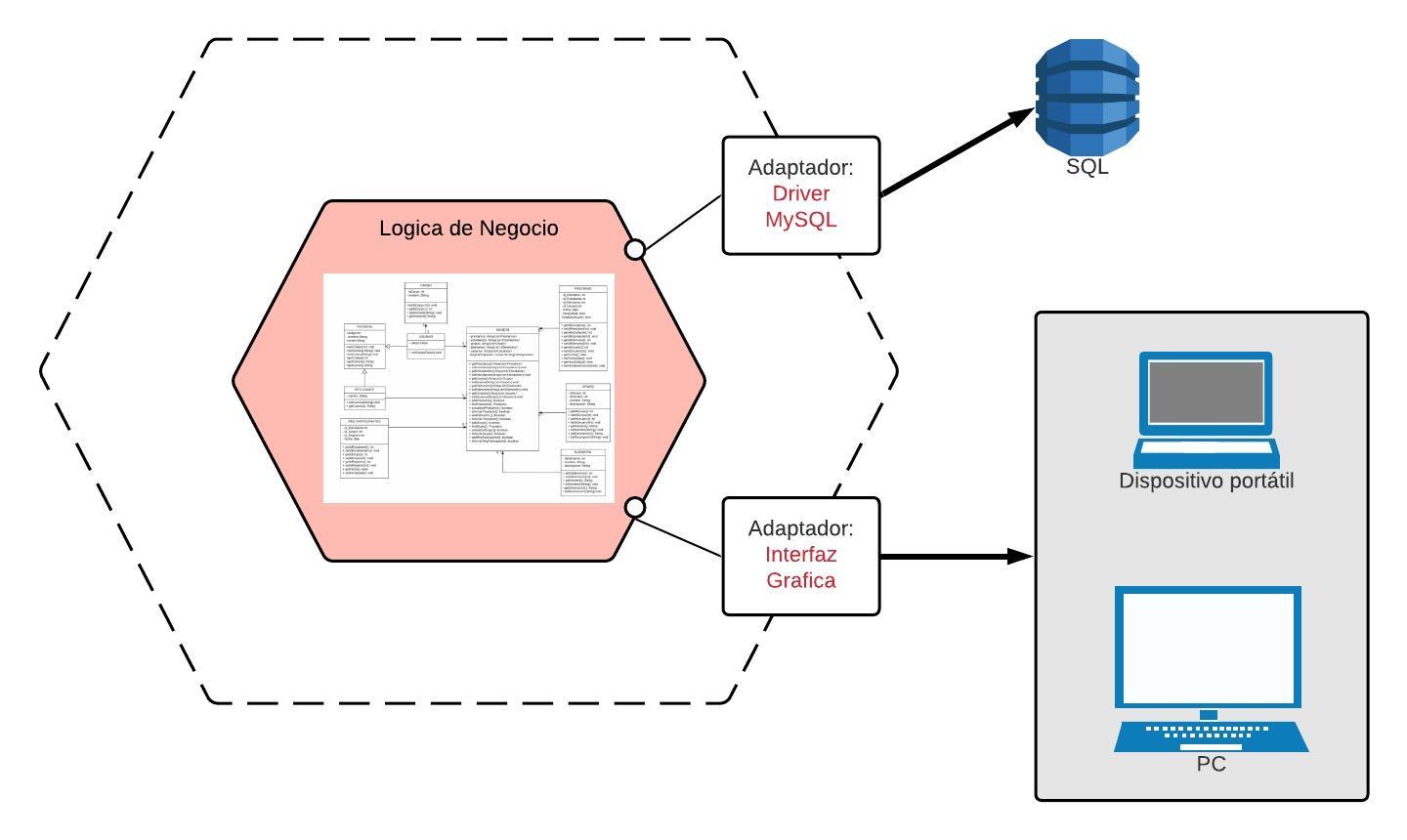
|  |  |
| --- | --- |
| **Atributo** | **Descripción** |
| **Adaptabilidad** | Al ser una aplicación web el sistema será capaz de ejecutarse desde cualquier navegador desde cualquier dispositivo. |
|  |  |

## 8.3) Patrones de arquitectura

Como ya se había mencionado para ese proyecto se tiene conceptualizado bajo una arquitectura de tres capas dentro de las cuales se encuentra la presentación, la lógica y los datos, por ello podemos hacer uso de un patrón hexagonal con la finalidad de poder manejar cada una de las capas casi de manera independiente y obtener sus beneficios, por ejemplo, poder implementar una mejora en las interfaces graficas de la aplicación sin el riesgo de comprometer la capa lógica o la capa de datos. Además de esto se tiene la ventaja de que es posible llegar a incrementar la funcionalidad de manera gradual sin llegar a afectar el funcionamiento de las partes ya implementadas, esto si así se requiere a futuro. Estas tres capas interactuaran entre si para que en conjunto puedan llegar a dar solución a nuestro problema planteado.

La capa de presentación contiene todo el funcionamiento que presentará la interfaz gráfica, para lo que se hará uso de un lenguaje propio de la interfaz y se incluirán todos los componentes necesarios para su funcionamiento iniciando con aquellos que permiten llevar la información y presentar las respuestas al cliente, lo que se definirá con más claridad posteriormente. La capa de datos como tal solo se encargara de contener las bases de datos y mantenerlas trabajando de la mejor manera posible y por último la capa lógica se encargara de contener todo el código que se refleja de los requisitos funcionales, además de ello también debe estar desarrollada de manera que se encuentre en disposición de recibir la información del usuario, resolver sus peticiones y devolver cierta información que la capa de presentación deberá interpretar para dar una respuesta legible y entendible al usuario, y además de esto, también debe estar en la capacidad de poder comunicarse con la capa de datos para hacer las peticiones respectivas y procesar las respuestas que posteriormente también serán comunicadas de alguna manera al usuario.

Para expresar esto de una manera más sencilla hacemos uso de una imagen dentro de la cual se reflejan los elementos que este patrón tiene, dentro de los cuales podemos encontrar los puertos y los adaptadores, la explicación se dará posterior a la imagen.



Para el caso, asumimos que la capa de dominio que se presenta en el patrón hexagonal es lo que nosotros denominamos como capa lógica, la cual esta dispuesta a ser accedida por los usuarios para poder llegar a realizar sus tareas de gestión es sus respectivos ámbitos, y además la capa lógica tiene un puerto abierto que le permite por medio de su correspondiente adaptador conectarse a toda la parte de bases de datos.

En primer lugar, la capa de presentación estará desarrollada en HTML apoyada en frameworks como Bootstrap y con elementos scripts de JavaScript para dar dinamismo y funcionalidad a las interfaces. Dentro de estas funcionalidades encontramos la verificación de la coherencia de los datos ingresados por el usuario y el envió de los mismos a la capa lógica haciendo uso del adaptador que para el caso puede estar dado por AJAX que es una técnica de desarrollo que mantiene comunicación asíncrona con el servidor.

En segundo lugar, la capa lógica tiene dispuesto un puerto para poder llegar a comunicarse con la capa de presentación con la finalidad de recibir información y devolver alguna respuesta, es importante mencionar que en un inicio se plantea reflejar toda la lógica de los requisitos en un lenguaje de alto nivel como JAVA. En tercer lugar, la capa lógica también tiene dispuesto un puerto para que haciendo uso de el adaptador pertinente pueda tener comunicación con la capa de datos, la cual tendrá contenidas bases de datos relacionales desarrolladas en MySQL por lo tanto el adaptador podrá estar dado por el uso de un driver que permita establecer una conexión y una sesión entre estas dos capas.

## 8.4) EventSourcing

Debido a que el sistema implementa una base de datos por medio de acciones CRUD y hace uso de un patrón de arquitectura hexagonal, este no hace uso del patrón EventSourcing para su funcionamiento.

## 8.5) Microservicios

Al ser un sistema pequeño, no requiere de muchos servicios para su funcionamiento. Además, se debe tener en cuenta que es un sistema simple, diseñado como ejercicio académico, por lo cual implementar un patrón de microservicios no es necesario.

## 8.6) Administración cliente servidor

**Cliente:** Para el caso del sistema el cliente se representa por los dispositivos desde los cuales accede el usuario por medio del navegador web. El navegador es quien se encarga de renderizar las vistas que dispone el sistema para la interfaz gráfica y por medio de las cuales el cliente realiza las peticiones requeridas al servidor

**Servidor:** El servidor en este caso es la base de datos de clever Cloud, a la cual se le envían las peticiones para acceder a la información contenida en ella, que permite hacer las gestiones de CRUD para los préstamos y los grupos de la Universidad.

## 8.4) Consumo web

Al ser un sistema manejado bajo el protocolo HTTP, el sistema de gestión de bases de datos hace uso de una REST, la cual, por medio de peticiones, get, post, put y delete hace posible el consumo de los datos disponibles para el sistema de gestión a crear.

## 9) Construcción

## 9.1) Lenguaje de construcción

Para la construcción del prototipo expuesto anteriormente se hará uso del lenguaje de programación Ruby; debido a que es un lenguaje fuertemente orientado a objetos que mediante el uso de gemas permite la extensión se su funcionalidad brindando a los desarrolladores la posibilidad de realizar los procesos de codificación de manera más sencilla, divertida y ágil. Además, una de las gemas que ofrece este lenguaje conocido como Rails permite la creación de páginas web, ofreciendo un framework fácil de usar que por medio de un patrón MVC hace uso de múltiples gemas y herramientas para realizar todo lo relacionado a la construcción de vistas, controladores y conexiones con la base de datos necesarias para el prototipo que se busca realizar.

## 9.2) Fundamentos de construcción

## 9.2.1) Biblioteca de construcción

|  |  |  |
| --- | --- | --- |
| **Código** | **Fuente** | **Uso** |
|  |  |  |
| <div class="row justify-content-center">  <div class="col align-self-center"> | [Columns · Bootstrap v5.0 (getbootstrap.com)](https://getbootstrap.com/docs/5.0/layout/columns/) | El código usado para la los contenedores de inicio con el que se buscaba centrar el contenedor en todo el centro de la ventana. |
| <table class="table">  <tr>  <th scope="col">#</th>  </tr>  </thead>  <tbody>  <tr>  <th scope="row">1</th>  <td>Mark</td>  <td>Otto</td>  <td>@mdo</td>  </tr>  </tbody>  </table> | [Tables · Bootstrap v5.0 (getbootstrap.com)](https://getbootstrap.com/docs/5.0/content/tables/) | Código usado para dar forma a las tablas que permiten hacer la función de listar dentro del sistema. |
| <input class="form-control form-control-lg" type="text" placeholder=".form-control-lg" aria-label=".form-control-lg example"> | [Form controls · Bootstrap v5.0 (getbootstrap.com)](https://getbootstrap.com/docs/5.0/forms/form-control/) | Código usado para darle forma a todos los campos necesarios para el sistema. |
| <button type="button" class="btn btn-secondary">Secondary</button> | [Buttons · Bootstrap v5.0 (getbootstrap.com)](https://getbootstrap.com/docs/5.0/components/buttons/) | Código utilizado para dar forma y color a los botones dentro del sistema. |
| #3D3C3A | [Códigos de Colores HTML (htmlcolorcodes.com)](https://htmlcolorcodes.com/es/) | Color usado para la barra |
| #ECC72F | [Códigos de Colores HTML (htmlcolorcodes.com)](https://htmlcolorcodes.com/es/) | Color usado para el activo de inicio y el contenedor donde se visualiza el nombre del usuario logueado y el cargo del mismo, en los módulos de préstamo y grupos. |
| rgba(24, 22, 21, 0.8) | [Códigos de Colores HTML (htmlcolorcodes.com)](https://htmlcolorcodes.com/es/) | Color u profundidad usada para el contenedor de inicio de sesión y registro de usuarios |
| Modelos, controladores, gemfile, device, layouts, | [(33) 1. Curso de Ruby on Rails 4 desde cero - Introducción - YouTube](https://www.youtube.com/watch?v=LMD3P97gXa0&list=PLpOqH6AE0tNiQ-ofrDlbAUSc1r67r_AWv) | Los códigos mostrados en estos videos sirven como base para la construcción básica de la aplicación. De allí se tomaron ideas para la validación, creación y desarrollo total de la aplicación, por lo tanto, es una ayuda de gran magnitud para futuros cambios que se quieran llegar a implementar y para entender más fácilmente la manera en la funciona Ruby on Rails. |

## 9.2.2) Minimizar complejidad

* + - **Técnicas de legibilidad**

**Programación estructurada:** Para el proceso de construcción del prototipo expuesto y en especial para la minimización de la complejidad se hará uso de la programación estructurada, basado en las funciones de Ruby, dentro de las cuales es posible definir las funciones del sistema por medio de bloques de código asociados a diferentes objetos del sistema; lo que permite que el código presentado sea mucho más ordenado y comprensible.

**Uso de etiquetas y ids:** En lo que respecta al proceso de construcción de las vistas, se hará uso de etiquetas div con contenedores dentro de las cuales se definirán los diferentes elementos de la vista, de manera que a nivel de interfaz el usuario pueda observar elementos ordenados y que sean llamativos; y a nivel de código sea sencillo identificar la utilidad de cada elemento brindándole nombres identificadores relacionados con su función.

* + - **Elementos de construcción**

Los elementos de construcción que será necesarios para hacer una correcta minimización y comprensión se basan en los principios basicos de Ruby on Rails de Don’t repeat yourself y convención sobre configuración; lo que nos guía a diseñar código por medio de los elementos explicados a continuación que permiten la reutilización y organización del código logrando que el equipo de desarrollo no se centré en la configuración del marco de trabajo sino que haga un enfoque al diseño del producto como tal.

**Clases:** El uso de clases es vital para la construcción del sistema, ya que estás permiten definir los elementos claves del sistema y las funciones respectivas de cada una. La idea principal es utilizar clases con una nomenclatura mayormente en ingles que haga relación al elemento al que se asocia la clase que cuenten con funciones especificas de este elemento, con el fin de evitar repeticiones innecesarias de código y que este a su vez sea entendible no solo por personas del entorno sino por personas a nivel global. Para el caso se harán uso de las clases de control y modelo, las cuales se usarán en el backend de la aplicación.

**Layouts:** Los layouts son estructuras creadas por rails para el manejo de las vistas, que funcionan como plantillas básicas de las vistas en donde se definen elementos comunes de las mismas. Para el caso se hará uso de tres layouts diseñados principales; el primero diseñado para todos los elementos de autenticación y registro de usuarios; el segundo para las vistas de gestión de grupos y el tercero para las vistas de préstamos.

**Funciones:** Las funciones son elementos que se encuentran dentro de las clases las cuales cumplen cada una con una función básica de la clase, dependiendo de cuál sea esta.

**Helpers:** Los helper son elementos que se pueden usar en el frontend, dentro de los cuales se pueden definir ciertos códigos de Ruby que permiten realizar interfaces dinámicas. En el prototipo se harán usos de estos elementos para mostrar algunos datos imprescindibles en la interfaz de usuario y para transportar la información de la vista al control.

* + - **Estructuras de control**

**If:** La estructura if se utilizará para realizar las validaciones de la entrada de datos de manera que no se envíen datos incongruentes a la base y s muestren los mensajes pertinentes tanto de éxitos como de error.

**For:** Este tipo de ciclos se verán reflejados en los campos de select del código, en donde deberán imprimirse algunos arreglos enviados desde la base de datos, para lograr que la interfaz sea más intuitiva y agradable al usuario.

* + - **Estructuras de gestión de fallos**

El sistema hace uso de técnicas de validación en cada uno de los módulos, por medio de estructuras if y alertas generadas mediante herramientas de Flash Messages, de manera que sea posible informar al usuario en caso de que se llegue a detectar errores o inconsistencias en los datos que digita el usuario. Lo que garantiza que el sistema cuide la integridad de los datos y le permita al usuario conocer en que está fallando para corregirlo a tiempo.

* + - **Prevención de errores**

En caso de existir errores de desbordamiento de memoria y demás fallos, el sistema informará del fallo y la ubicación del mismo; pero permitirá el funcionamiento del resto del sistema. Al ser un prototipo construido con una bajo acoplamiento será posible que el usuario continué la ejecución de otras funcionalidades, mientras se repara aquella que se encuentra fallando.

* + - **División Jerárquica**

**Paquetes:** Debido a que el proyecto se encontrará basado en un patrón MVC este se divide en tres grandes paquetes: El modelo, la vista y el controlador dentro de los cuales se encontrarán los diferentes paquetes y elementos necesarios para su función; como se muestra en el diagrama de paquetes presentado en la función 7.2 del documento

**Clases y funciones:** Se pretende manejar tanto el nombre de las clases como el nombre de las funciones acorde a los objetos que representan y las responsabilidades que tiene asignadas, las cuales se presentan en el diagrama de clases presentado en la imagen 15. De la sección 7.1 del documento.

* + - **Afinación de código**

La afinación del código se hará por medio de pruebas tanto formales como informales, de manera que sea posible detectar los errores que se vayan encontrando y corregirlos a tiempo. Además, el sistema cuenta con variables con valores consistentes, de manera que no se presenten ambigüedades a la hora de usarlo.

## 9.2.3) Anticipación de cambios

* + - **Elementos sensibles a cambios**

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| **Interfaz Grafica** | Al ser una interfaz sujeta a las características presentadas por la presentación de los demás portales web de la Universidad; este elemento va a variar de acuerdo a como se actualicen estos componentes virtuales de la institución. |
| **Bases de datos** | Debido a que el sistema propuesto es un prototipo realizado como ejercicio académico, la base de datos utilizada no se encuentra diseñada para cumplir todos los requerimientos que demandan los demás sistemas de la Universidad; por lo cual es importante que si en algún momento se llega a evolucionar el programa para una implementación real, se haga cambio a una base de datos más adecuada. |
|  |  |

* + - **Elementos que pueden tener cambios**

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| **Modulo Participantes** | El modulo participantes se encuentra compuesto con la opción de crear; sin embargo, no es un modulo que cuente con todas las acciones de Crud que le corresponden; por lo cual para un futuro y de verse necesario se puede llegar a generar un controlador específico para este módulo, dentro del cual se incluyan las demás funciones faltantes |
| **Modulo Inventario** | El modulo inventario se toma como un submodulo del módulo de préstamos, por lo cual realizar todas las acciones de Crud le pertenecen al módulo de préstamos; En caso de ser necesario y con tal de general una cohesión más alta y un bajo acoplamiento, se puede llegar a generar un controlador especifico para esta función con todas las operaciones que requiere. |
| **Controladores existentes** | Al ser un prototipo diseñado como ejerció académico y que puede llegar a evolucionar más adelante para una implementación real, todos los controladores creados pueden llegar a variar eliminando o colocando más funcionalidades que hagan la experiencia del usuario más cómoda. |

* + - **Elementos que no cambian**

|  |  |
| --- | --- |
| **Elemento** | **Descripción** |
| **Modelos** | Los modelos generados para el sistema se encuentran diseñados de manera que puedan soportar los cambios mencionados anteriormente, por lo que no existe un cambio justificable para los mismos |
| **Inicios de sesión** | Los controladores de inicio de sesión están diseñados con todas las validaciones y herramientas que garantizan la seguridad de la información, por lo que no son elementos que requieran o puedan requerir un cambio. |

* + - **Construir para verificar**

|  |  |
| --- | --- |
| **Elemento** | **Descripción de la verificación** |
| **Base de datos** | La base de datos al ser un elemento externo, que depende de un servidor puede llegar a presentar ciertos fallos; Por lo cual es necesario hacer pruebas tanto de rendimiento, como en el funcionamiento de la misma y generar las tolerancias a fallos necesarias para soportar dichos errores. |
| **Funciones** | Las funciones son la clave del sistema, por tanto, es importante que durante todo su proceso de construcción se realicen pruebas unitarias por medio de herramientas de Rails como Minitec; además de generar las pruebas de rendimiento respectivas para cada función usando la gema rack-mini-profile. |
| **Servidor** | El servidor es uno de los puntos más debiles del sistema; al ser un equipo o maquina normal, puede que este llegue a generar problemas de cuellos de botella o rendimiento lento del sistema. Sin embargo, al ser un prototipo que no se va a llegar a implementar en su totalidad, estos problemas puede que no se lleguen a evidenciar; no obstante, es importante realizar las pruebas de rendimiento necesarias con la gema rack, con el fin de encontrar los cuellos de botella que se puedan generar o el lento funcionamiento del servidor. |

## 9.3) Repositorios

Para el desarrollo de este proyecto se hizo uso un repositorio creado en GitHub disponible en el link:

<https://github.com/carlossalamanca02/ProyectoSoftwareII.git>

Allí, se evidencian los cambios realizados sobre el proyecto teniendo en cuenta que estos se realizaron acorde a las fechas planeadas desde un inicio en el cronograma de actividades. Por lo cual, las fechas en las cuales se presentaron cambios en el repositorio van como sigue a continuación:

* Creación del repositorio: 07/04/2021
* Login: 08/04/2021 – 12/04/2021
* Modulo de grupos: 14/04/2021 – 08/05/2021
* Módulo de préstamos: 03/05/2021 – 10/05/2021
* Pruebas unitarias: 10/05/2021 – 14/05/2021
* Documentos: Se realizan de manera transversal

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Abril | | | | Mayo | | | |
| **Actividades** | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1. Creación del repositorio |  |  |  |  |  |  |  |  |
| 1. Módulo de grupos |  |  |  |  |  |  |  |  |
| 1. Módulo de prestamos |  |  |  |  |  |  |  |  |
| 1. Pruebas unitarias |  |  |  |  |  |  |  |  |
| 1. Documentos |  |  |  |  |  |  |  |  |

## 9.4) Gestión de la configuración

## 9.4.1) Planificación de construcción

|  |  |
| --- | --- |
| **Elemento** | **Método de construcción** |
| **Base de datos** | Debido a que el sistema que se va a implementar el prototipo de un sistema pequeño, que realmente no se va a llegar a implementar y teniendo en cuenta el conocimiento que se tiene actualmente sobre bases de datos; se hace pertinente hacer uso de una base de datos relacional en SQL, haciendo uso de hosting gratuitos como clever Cloud. Que permitan mantener la base de datos en un servidor al cual se puede acceder desde cualquier punto sin problemas. |
| **Lógica del negocio/ API** | Para la lógica del negocio, que en este caso funcionará como una especie de API, se hará uso de código Ruby unido a las propiedades de convención sobre configuración de Raisl, de manera que la implementación de los métodos de CRUD de cada modulo se puedan generar en el respectivo controlador y que desde allí la conexión a la base sea muy sencilla y rápida de implementar. |
| **Interfaz Grafica** | La interfaz grafica se va a realizar por medio de código HTML, junto con Bootstrap para temas de maquetación; de manera que se puedan construir interfaces organizadas y llamativas a la vista del usuario. Además, por medio de extensiones erb se busca conseguir no solo interfaces llamativas sino también funcionales, que tengan acceso rápido y fácil a los controladores para el transporte de información. |

## 9.4.2) Métricas de construcción

|  |  |
| --- | --- |
| **Métrica** | **Explicación** |
| **Código desarrollado** | Debido al principio de convención sobre configuración que maneja Ruby on rails y las herramientas que ofrece en cada una de las capas como lo es Active Record, Action Controller y Action view, es posible afirmar que el porcentaje de código que se desarrollo como tal por parte de los desarrolladores del sistemas es del 50%; ya que las herramientas previamente explicadas ofrecían una sistema codificado que permite realizar conexiones; renderizaciones y en general todo el transporte de la información a lo largo del sistema que disminuían en su gran mayoría el trabajo de codificación y configuración. |
| **Código reutilizado** | Teniendo en cuenta que el código reutilizado se encuentra principalmente en el paquete de vistas del sistema es posible decir que aproximadamente el 30% del código a sido reutilizado; lo que es claramente visible en algunos documentos en donde la cantidad de código utilizado no supera las 10 líneas de código; ya que el resto se encuentra diseñado en un documento que sirve para todos. |
| **Código destruido** | El porcentaje de código destruido en la elaboración del sistema fue aproximadamente del 40%; debido a que si bien se tenia cierto conocimiento de Rails, al ser un proyecto realizado por estudiantes en formación, no se puede decir que se tuviera un conocimiento total de como llegar a generar un proyecto como el que se está implementando actualmente; por lo cual fue necesario ir probando algunas herramientas y que hicieran el trabajo más fácil y que minimizaran la complejidad del código desarrollado al mínimo. |
| **Inspección de código** |  |
| **Errores encontrados** | El porcentaje de errores encontrados durante todo el proceso es aproximadamente de un 45%; debido a que al ir aprendiendo sobre la marcha como se espera que se haga en un ejercicio académico; resulta un poco difícil codificar sin tener un porcentaje de errores encontrados y corregidos. |

## 9.5) Reutilización de código

|  |  |
| --- | --- |
| Elemento | Descripción de la reutilización |
| Barras de navegación | Las barras de navegación necesarias para cada uno de los módulos, es decir para préstamos y grupos es común para todas las interfaces de sus sub módulos; por lo tanto por medio de los layouts o plantillas de rails es posible hacer la creación de cada una de las respectivas barras y por medio de funciones render implementadas en el controlador de cada módulo, lograr que dicha interfaz se implemente para todas aquellas interfaces de los sub módulos |
| Tablas de búsqueda | Las tablas de búsqueda o lista de elementos, ya sea de grupos para el caso del módulo de grupos o de préstamos para el módulo de préstamos; es un código e interfaz común que tienen las funciones de buscar, modificar y eliminar en cada módulo. Por lo cual, por medio de plantillas parciales, es posible crear un documento que contenga el código respectivo e implementarlo en la parte de la interfaz que se desee con etiquetas de tipo render. |
| Inicio modulos | La imagen de inicio de ambos módulos es la misma, por tanto, su reutilización se hará por medio de plantillas parciales, que permitan renderizarse en donde se requiere |
| Cabeceras y Pies de pagina | Las cabeceras y pies de pagina son comunes para todas los sub módulos del módulo de préstamos y del módulo de préstamos; Por lo cual al igual que en las barras de navegación su implementación se realizará por medio de layouts que permitan reutilizar el código en las vistas necesarias. |
| Módulo de login | El fondo y contenedores del modulo de inicio son comunes para la vista de login y registro de usuarios, de manera que es posible reutilizar dicho código por medio de layouts que rendericen la interfaz común deseada. |

## 10) Pruebas de verificación

## 10.1) Pruebas de base de datos

|  |  |
| --- | --- |
| **Prueba** | **Descripción** |
| **Pruebas de modelos** | El funcionamiento de los modelos generados por rails para la administración de la base de datos se pueden llegar a probar por medio de las gemas de testeo de raisl, es decir selenium-driver y capybara; ejecutando pruebas para la creación, busqueda, actualización y eliminación de los elementos en cada una de las tablas. Además, se llegarán a probar la validación de los datos, para asegurar que en caso de que se envien datos inconsistentes no se haga la inserción o consulta respectiva. Todo aquello se encontrará disponible en el ditectorio /test/modelos/ dispuesto por rails, y para no afectar el funcionamiento de la base generada para la producción y el desarrollo se hará uso de una base de datos diferente llamada test, que haga uso del mismo esquema y por ende los mimos modelos. |
| **Pruebas de rendimiento** |  |

## 10.2) Pruebas de utilidad

|  |  |
| --- | --- |
| **Etapa** | **Descripción** |
| **Hipótesis** | * + **Claridad en mensajes de validación:** Para que el cliente se sienta a gusto con la aplicación, en primer lugar, se debe verificar que todos los mensajes de validación se encuentren funcionando de la mejor manera y sean claros para el usuario. Comprobando que los mansajes se muestren en donde se deben, sean concisos, no tengan errores de semántica u ortografía; etc.   + **Verificación de campos:** Para que el cliente se sienta a gusto con la aplicación se debe verificar que su percepción sobre los formularios sea positiva; es decir que entienda que información debe ir en cada campo y como debe digitarla; además se debe verificar que los campos estén validen los datos ingresados de manera que no se presenten errores de inconsistencia en la base.   + **Botones:** Otro de los aspectos importantes a tener en cuenta son los botones que maneja la aplicación; es decir que en cada vista el usuario entienda de forma intuitiva como enviar la información o como limpiar los campos. Además, se debe revisar que la ubicación de dichos botones resulte cómoda al usuario.   + **Barra de navegación:** Es importante comprobar que la barra de navegación funciona para el usuario, es decir que los iconos, nombres y demás sean claros para este y que a si mismo los vínculos se encuentren funcionando correctamente. |
| **Métricas** | * **Objetivas:** En este ítem se evaluará la cantidad de errores que el usuario comete a la hora de ingresar la aplicación en los campos de manera que se pueda determinar si son o no son claros para el usuario. Además, se tendrá encuentra la tasa de éxitos que se obtenga en cada una de las acciones que realice el usuario y la tasa de errores. * **Subjetivas:** En este ítem se evaluará el nivel de confusión o estrés que tenga el usuario con la aplicación por medio de encuestas que permitan determinar cual fue la percepción del usuario frente a la aplicación; teniendo en cuenta aspectos como el esfuerzo percibido, el cumplimiento de las expectativas; entre otros. De igual manera mientras este se encuentre interactuando con la aplicación se le harán algunas preguntas que permitan determinar como se siente con lo que ve y con lo que sucede con la app. |
| **Tareas y escenarios** | Para realizar estas pruebas se invitará al usuario a hacer uso de la aplicación. En un inicio se le darán indicaciones de lo que debe hacer en caso de que este las solicite. En primer lugar, el usuario debe registrarse para hacer la autenticación respectiva. Una vez se haya a logueado el usuario iniciará la creación y gestión de elementos a su convencía; la idea es que este vaya contando como le parece la aplicación, que le gusta y que le disgusta. |
| **Reclutamiento** | Debido a que este proyecto se encuentra realizado como ejercicio académico los usuarios serían los mismos estudiantes que diseñaron el programa, de manera que no se tendrá en cuenta esta etapa para el caso. |

## 10.3) Pruebas Estáticas y dinámicas

|  |  |
| --- | --- |
| **Pruebas** | **Descripción** |
| **Estáticas** | Dentro de las pruebas estáticas se va a realizar una revisión detallada de todo el código, documento por documento, línea por línea, en busca de errores de identación, código amontonado, falta de estructura, puts o impresiones por consola que se haya olvidado borrar o errores de sintaxis que puedan generar fallos en la aplicación. Esto no solo con el fin de corregir posibles errores, sino también de hacer el código más ordenado, entendible, manejable y mantenible. |
| **Dinámicas** | Las pruebas dinámicas se realizarán en primer lugar de forma manual y empírica, ejecutando el servidor y probando que cada uno de los módulos, funciones, campos, botones, validaciones, redirecciones y en general cada parte del código funcione de manera correcta. Además, también se realizarán pruebas de rendimiento por medio de aplicaciones como jmeter; y pruebas de usabilidad en donde el usuario interactúe directamente con el sistema. |

## 10.4) Pruebas Modernas

Las pruebas modernas implemententadas para el proyecto se van a realizar a través de pruebas automatizadas, haciendo uso de las gemas de testing implementadas por Rails Selenium driver y capybara. Estas dos extensiones permiten la generación de ambientes de pruebas sencillos, generando documentos de testeo para los modelos y controladores. Además, ofrece una serie de helpers que permiten se encargan de realizar la simulación de un usuario, para comprobar que todo esté funcionando de manera esperada.



Imagen 1. Logos re Rails y capybara

## 10.5) Repositorio de administración de la configuración

Una vez el sistema sea entregado se establecerá este punto como línea de referencia, este dispondrá de un directorio en donde se llevará el registro de todos los cambios que se lleguen a implementar en el sistema luego de dicha línea. Cada vez que se vaya a ejecutar un cambio el desarrollador debe acceder al repositorio del proyecto, ir al directorio “/ACS/” disponible dentro del proyecto y en una rama diferente a la principal generar una carpeta de la siguiente manera:

* Nombre con formato: Cambio\_N°\_Año.
* Dentro de la carpeta se debe generar un archivo con la documentación respectiva del cambio, dentro de las cuales se debe reportar todo el proceso de cambio con las diferentes fases del mismo, es decir: Reconocimiento de la necesidad del cambio, valuación de la necesidad del mismo y la viabilidad por parte del desarrollador, reporte del cambio, respuesta por parte de la auditoria, etc.
* Además de debe documentar todo el proceso de pruebas del cambio, de manera que se valide su funcionalidad y la respuesta positiva del sistema al implementarlo.
* En caso de que el cambio sea aprobado se debe incluir en que versión del código se implementó y la fecha en la que fue aprobado el mismo.

## 10.7) Administración de cambios

Durante el proceso de desarrollo del sistema y luego de la entrega del mismo se hace necesario manejar una administración de cambios, para evitar que los integrantes del grupo o cualquier persona que quiera cambiar alguna parte del sistema lo haga sin el conocimiento y aprobación de los otros; por tanto, es necesario que a la hora de especificar un cambio se implemente el siguiente procedimiento.

1. Solicitud de cambio: Cada participante que requiera implementar un cambio no planeado con anterioridad debe informarle al otro integrante de manera que ambos tengan conocimiento de lo que se está haciendo y el por qué.
2. Aprobación: Si se requiere implementar un cambio ambos integrantes del grupo deben evaluar la pertinencia del cambio y en caso de ser aprobada haciendo uso de programación en parejas será implementada.
3. Desarrollo: Los cambios desarrollados deben ser adecuadamente probados antes de ser enviados al repositorio se implementen en a la rama principal del mismo.
4. Implementación: Una vez se valide y verifique el cambió implementado se debe hacer el commit respectivo en el repositorio con un mensaje descriptivo de los cambios realizados.
5. Documentación: De ser un cambio realizado dentro del proceso de desarrollo la documentación se debe añadir luego de la verificación del cambio en el documento respectivo. De ser un cambio realizado después de la fecha de entrega se debe realizar la documentación respectiva y añadir el cambio al repositorio de cambios con las especificaciones respectivas.
6. En el reporte del cambio se debe implementar la descripción de las siguiente 4 clases:

|  |  |
| --- | --- |
| **Clase** | **Descripción** |
| **Alcance** | El alcance de cada uno de los cambios no debe ser mayor que un módulo, debido a que ejecutar un cambio tan grande puede llegar a generar complicaciones para los usuarios. |
| **Roles** | Los cambios pueden ser reconocidos e informados por los usuarios, pero solo aprobados, desarrollados e implementados por los desarrolladores quienes a su vez hacen el papel de líderes. |
| **Clasificación** | Un cambio puede llegar a ser permanente, temporal o de emergencia; En lo que respecta a los cambios permanentes y temporales se deben analizar bajo las métricas definidas en los puntos anteriores; pero en el caso de los cambios de emergencia deben llevar un proceso más rápido y acorde a la necesidad del cambio a implementar. |
| **Implementación** | Dentro de la implementación del cambio se debe hacer una definición del mismo, una evaluación de los riesgos que puede llegar a generar, se debe aprobar como se definió en los ítems anteriores, luego de debe añadir al sistema y por ultimo se debe realizar sobre el mismo las respectivas pruebas y control d funcionamiento y eficacia para poder llegar a cerrar el proceso. |

## 10.8) Pruebas y anexos

## 11) Métricas de diseño para webapps

|  |  |
| --- | --- |
| **Métricas de interfaz** | |
| **Métrica** | **Cantidad** |
| Complejidad de plantilla | 3 |
| Complejidad de reconocimiento | 2 |
| Complejidad de región de plantilla | 5 |
| Complejidad de selección | 4 |

|  |  |
| --- | --- |
| **Métricas de estéticas** | |
| **Métrica** | **Cantidad** |
| Conteo de palabra | 50 |
| Tamaño de pagina |  |
| Conteo de color | 5 |
| Conteo de vínculos | 6 |

|  |  |
| --- | --- |
| **Métricas de contenido** | |
| **Métrica** | **Calificación** |
| Complejidad de pagina | 1 |
| Complejidad gráfica | 2 |
| Complejidad de audio | 0 |
| Complejidad de animación | 0 |

|  |  |
| --- | --- |
| **Métricas de navegación** | |
| **Métrica** | **Calificación** |
| Complejidad de vinculación de pagina | 2-6 |
| Complejidad de conectividad | 7 |

## 12) Estándares

* IEEE STD 830-1998 IEEE RECOMMENDED PRACTICE FOR SOFTWARE REQUIREMENTS
* IEEE STD 1063-2001 IEEE STANDARD FOR SOFTWARE USER DOCUMENTATION
* IEEE STD 1471-2000 IEEEE RECOMMENDED PRACTICE FOR ARCHITECTURAL DESCRIPTION OF SOFTWARE INTENSIVE
* IEEE STD 1012-1998 IEEE STANDARS FOR SOFTWARE VERIFICATION AND VALIDATION

## 13) A tener en cuenta

Como se ha mencionado a lo largo del documento, este sistema es un prototipo de un sistema web que permite la Gestión de grupos culturales y prestamos de elementos culturales de la UPTC seccional Sogamoso; por lo tanto, es un sistema que puede estar sujeto a muchos cambios para llegar a implementarse de manera real como sistema para la universidad.

En primer lugar, es importante mencionar que el sistema no cuenta con funciones de CRUD completas para algunos elementos claves como los participantes de los grupos o los elementos de préstamos de la universidad. Acciones como crear y listar se encuentran presentes, pero las de actualizar y eliminar aún deben añadirse. Por otro lado, si realmente se quisiera implementar como sistema para la universidad el acceso al sistema se deberá realizar exclusivamente para usuarios que tengan el correo de la Universidad, haciendo necesarios cambios de bases de datos y servidores.

## 14) Conclusiones

Si bien es cierto que en nuestra formación como ingenieros en sistemas en un principio no se explica la necesidad de llevar un proceso de planificación, construcción e implementación de un sistema de software, a medida que se van adquiriendo más conocimiento y se va profundizando en los aspectos relevantes de la carrera se hace notable la importancia de estos procesos para la construcción de sistemas de calidad que realmente ayuden a la comunidad y permitan avanzar y facilitar la realización de diferentes autorizadas.

El sistema explicado en este documento, a pesar de ser construido como prototipo, permitió que como estudiantes entendiéramos múltiples conceptos y procesos necesarios para construir e implementar software de calidad. Como resultado de este trabajo se definió una arquitectura de tres capas basada en fundamentos de calidad de software definidos por los estándares de calidad respectivos y implementó la metodología ágil XP que con ayuda de la programación en parejas permitió la debida construcción de un prototipo para un sistema de gestión de grupos y prestamos pensado para la Universidad Pedagógica y Tecnológica de Colombia seccional Sogamoso.

## 15) Bibliografía

* [Apache JMeter - Apache JMeter™](https://jmeter.apache.org/)
* [4 (uah.es)](http://www.cc.uah.es/drg/b/HispaSWEBOK.Borrador.pdf)

## 16) Infografía

1. [(33) 1. Curso de Ruby on Rails 4 desde cero - Introducción - YouTube](https://www.youtube.com/watch?v=LMD3P97gXa0&list=PLpOqH6AE0tNiQ-ofrDlbAUSc1r67r_AWv)
2. [Bootstrap · The most popular HTML, CSS, and JS library in the world. (getbootstrap.com)](https://getbootstrap.com/)
3. [Códigos de Colores HTML (htmlcolorcodes.com)](https://htmlcolorcodes.com/es/)

## 

## 

## 

## 

## 

## 

## 

## 

## 