Imagen que contiene Diagrama

Descripción generada automáticamente

Universidad de los Andes

Ingeniería de Sistemas y Computación

Diseño y Programación Orientada a Objetos

**Proyecto1**

Entrega 3- Documento de diseño final

Juan Diego Cruz Vega

[j.cruzv@uniandes.edu.co](mailto:j.cruzv@uniandes.edu.co) 202015077  
  
Mateo López Tobón  
[m.lopez24@uniandes.edu.co](mailto:m.lopez24@uniandes.edu.co) 202021416  
  
Mateo Cote Canal  
[m.cotec@uniandes.edu.co](mailto:m.cotec@uniandes.edu.co) 202022609

Bogotá, Colombia

**Principales Responsabilidades:**

1. Proyecto🡪 contiene los siguientes atributos necesarios para el funcionamiento de la aplicación:

* String: nombre🡪nombre del proyecto
* String: descripción🡪 descripción del proyecto
* String: fechaInicio🡪 contiene la fecha de inicio del proyecto.
* String: FechaFin🡪 contiene la fecha de finalización del proyecto.
* Participante: ParticipanteJefe🡪 nombre del participante líder del proyecto

1. Participante🡪 contiene los siguientes atributos necesarios para el funcionamiento de la aplicación:

* String: nombre🡪contiene el nombre del participante.
* String correo🡪contiene el correo del participante.
* String contraseña🡪 contiene la contraseña del participante.
* String UserName🡪 contiene el nombre de usuario del participante.

1. RegistroActividad🡪 contiene los siguientes atributos necesarios para el funcionamiento de la aplicación:

* Int Id🡪 contiene el id de la actividad.
* String titulo🡪contiene el titulo de la actividad.
* String tipo🡪 contine el tipo de la actividad.
* String fecha🡪 contiene la fecha de creación de la actividad.
* String HoraInicio🡪contiene la hora de inicio de la actividad.
* String HoraFin🡪 contiene la hora de finalización de la actividad.
* Participante Participante🡪 contiene el nombre de el o los participantes en la actividad.
* String tiempoC🡪 contiene el tiempo cronometrado.

**Diagramas.**

1. **Modelo de dominio**

Diagrama, Tabla

Descripción generada automáticamente

1. **Diagrama con atributos y sus métodos**

**Diagrama, Esquemático

Descripción generada automáticamente**

1. **Modelo del mundo.**

**Diagrama

Descripción generada automáticamente**

**Relaciones entre los objetos: (según los diagramas podemos describir las relaciones entre cada clase)**

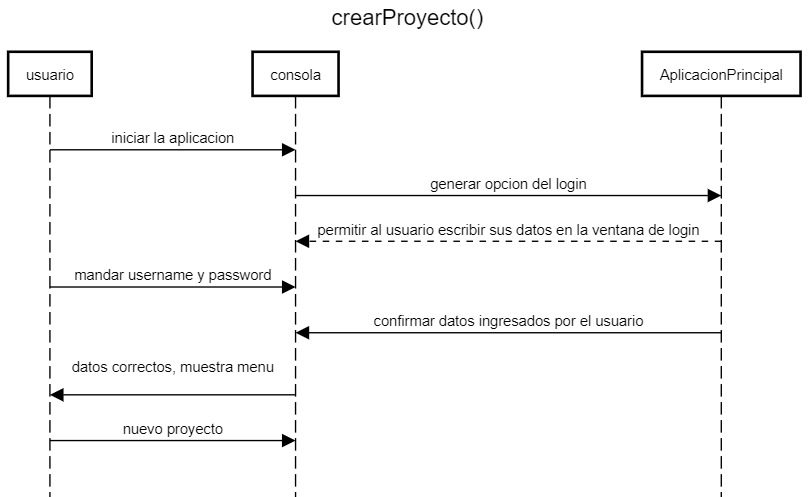
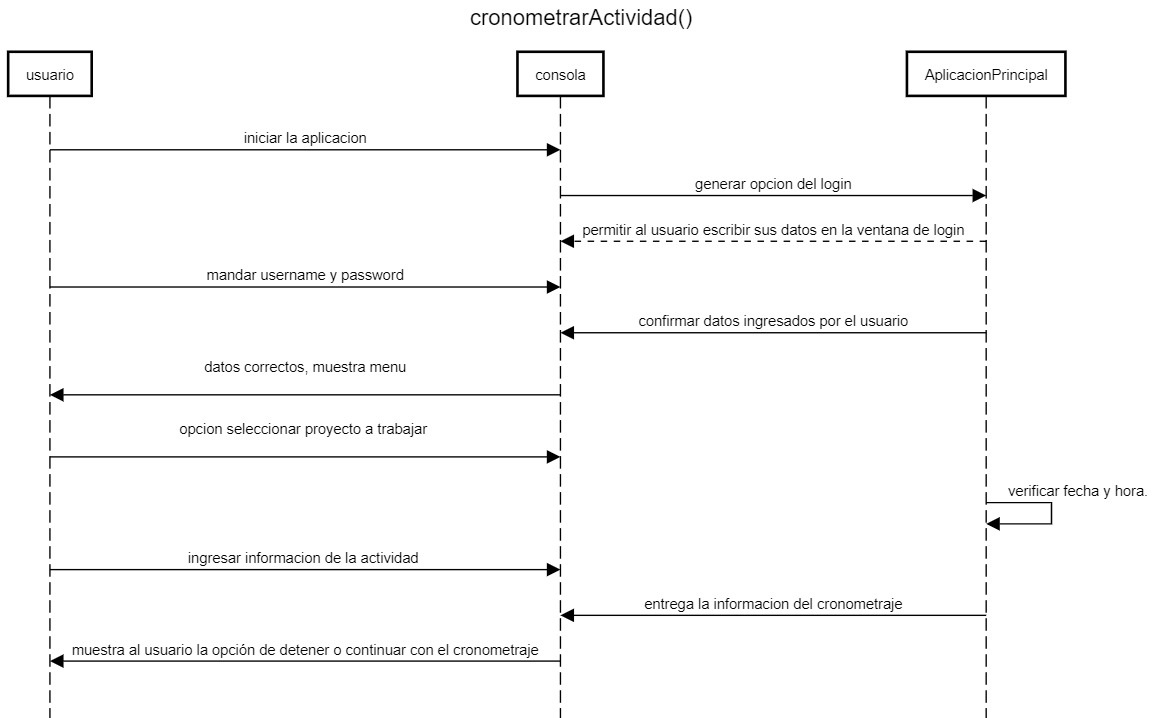
* **RegistroActividad->proyecto** : Proyecto este compuesto por actividades y del mismo modo puede ser modificado por estas, en proyecto puede haber una o muchas actividades.
* **aplicacionPrincipal–proyecto**: Por medio de la app principal se agregan proyectos y le ejecuta las funciones a este, por ejemplo, se le registran actividades a un proyecto específico.
* **AplicaciónPrincipal–>consola:** La aplicación principal implementa la consola, a través de esta se ejecuta la aplicación y se muestra el menú.
* **AplicaciónPrincipal–>registroActividad:** Por medio de la app principal se modifican y agregan los datos del registro de una actividad.
* **Participante–>proyecto:** Cada participante está en uno o más proyectos, haciéndolo proyecto un atributo de la clase participantes.

* **Participante–AplicacionPrincipal:** A través de la app principal se registran y modifican los datos de la clase participante.
* **Participante–>RegistroActividad:** Al igual que en reporteActividades, el participante realiza un registro de actividad, y provee los datos necesaria para esta.
* **RegistroActividad-**>**Tiempo:** La clase actividad hace uso de la clase Tiempo.
* **AplicacionPrincipal->Tiempo**: aplicación principal implementa la lógica con los atributos de Tiempo para cronometrar las actividades.

**Métodos empleados:**

1. **crearProyecto():** a través de la app principal se crea uno o varios objetos de la clase Proyecto, y se le dan sus valores respectivos.
2. **registrarActividad():** a través de la app principal se crea uno o varios objetos de la clase registrarActividad, y se le dan sus valores respectivos.
3. **modificarDatos():** a través de la app principal se modifican los datos existentes de una actividad o participante.
4. **ResporteParticipante():** dados los datos recolectados en los métodos anterior, la app principal crea un reporte con la información requerida de cierto psrticipante.
5. **registraUsuario():** a través de la app principal se crea uno o varios objetos de la clase Participante, y se le dan sus valores respectivos.
6. **login():** El usuario inicia sesión para poder realizar cualquiera de los métodos anteriores.
7. **menuProyecto():** muestra al usuario el menú con las opciones del proyecto.
8. **menuActividad():**muestra al usuario el menú con las opciones de cada actividad.
9. **AgregarParticipante():** permite al usuario agregar un participante al proyecto si así lo desea.
10. **modificarDatosProyecto():** Permite al usuario modificar los datos de un proyecto.
11. **AgregarActividad():** permite al usuario agregar una nueva actividad a su proyecto.

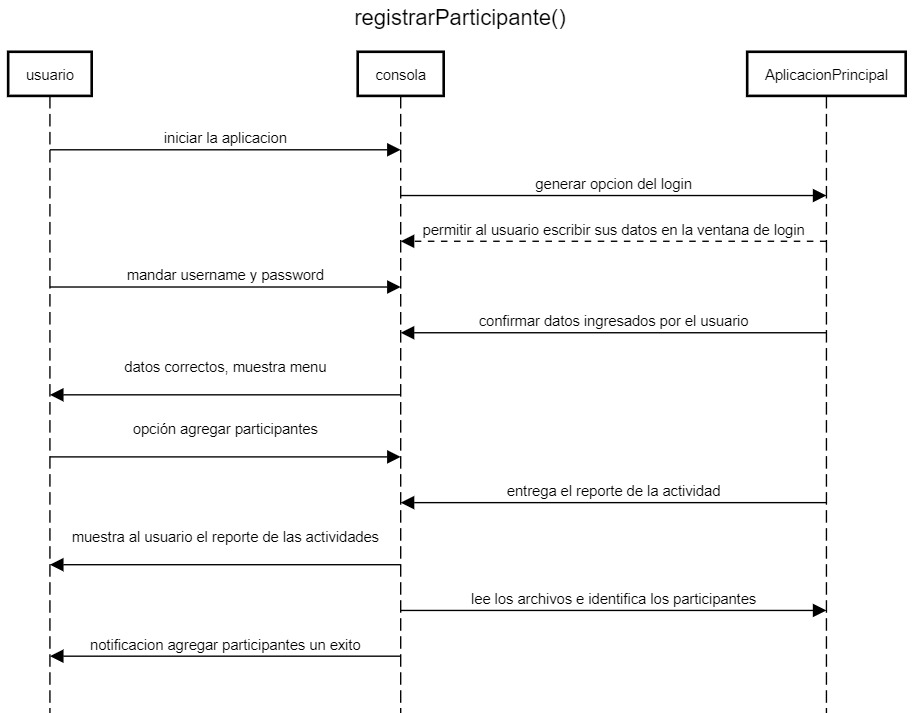
**Diagramas de secuencia**

* crearProyecto():  
  
* cronometrarActividad():   
  
* login()  
  Diagrama

  Descripción generada automáticamente
* zmodificarDatos()  
  Diagrama

  Descripción generada automáticamente
* mostrarReporteActividad()Diagrama

  Descripción generada automáticamente
* registrarActividad()Diagrama

  Descripción generada automáticamente
* registrarParticipante()

**Restricciones:**

El al formato de los archivos(csv) representa un ligero problema al momento de digitar los datos que la aplicación le pide al usuario, lo que sucede es que nuestra aplicación es estricta con las entradas, es decir que cualquier dato mal digitado representara un error que sacara al usuario de la aplicación y le tocara hacer todo el proceso desde cero, esta restricción causa bastante confusión en métodos como el de cronometrado ya que el formato debe ser respetado en su totalidad.

**Reflexiones**

Para esta iteración del documento de diseño se realizaron algunos cambios en el diseño de la aplicación ya que los consideramos pertinentes para la correcta organización y funcionamiento de nuestra aplicación, estos cambios fueron:

1. quitamos la clase ReporteActividades y la pusimos como un método de la aplicación principal.
2. Creamos una clase Tiempo para implementar el método que cronometra las actividades
3. Agregamos un método de login para que la aplicación tuviera un modo de autenticación para los usuarios, sobre todo si es participante líder.
4. Así mismo agregamos los métodos menuActividad, menuProyecto, AgregarParticipante, y AgregarActividad.

**Observaciones adicionales.**

Decidimos implementar la aplicación en dos paquetes, uno para las clases principales y el otro contiene la lógica de la aplicación y la consola, con esto en mente implementamos todos requerimientos funcionales en una clase llamada aplicación principal y la clase consola solo preguntamos al usuario si desea crear una cuenta o si ya esta registrado con una, una vez identificados estos datos el programa empieza a correr los métodos implementados en la aplicación principal.