****

**“**PROYECTO TAD”

Cuentas Claras, Amistades Largas

sdij

Cuen

**Fernando Rámila Gómez, Naiara Landeta González Cayetano Romero Vargas**

**1º Edicion**

**Índice**

**Problema a resolver** 3

**Requerimientos del problema** 4

**Esquema funcional** 5

**Diagrama UML** 6

**Implementación** 8

**Esquema Maven: Diagrama de Dependencias** 9

**Diagrama MongoDB** 9

**Manual de uso** 10

**Extras** 10

# **Problema a resolver**

El proyecto ‘Cuentas Claras, Amistades Largas’ es una aplicación orientada para dispositivos móviles, que permite llevar las cuentas de un usuario y sus amistades a lo largo de distintos eventos como viajes, fiestas, etc.

La aplicación permite llevar un control de los gastos de todos los participantes en el evento y una vez cerrado este, generará un resumen de las cuentas informando a cada participante de cuánto debe y a quién.

Consistirá en un conjunto de pantallas relacionadas entre sí, que gestionarán los distintos elementos de la aplicación (usuarios, amistades, eventos, pagos) y la información recopilada será almacenada en una base de datos.

La aplicación deberácumplir con los siguiente objetivos:

* Identificarse con un usuario y acceder al sistema.
* Gestionar eventos.
* Gestionar participantes.
* Gestionar un gastos.
* Gestionar usuarios.

# **Requerimientos del problema**

Para cumplir todos los requerimientos del problema se diseñan una serie de pantallas conectadas entre sí que permitan cumplir los objetivos citados en el punto anterior.

Las pantallas diseñadas tendrán que cumplir los siguientes requisitos:

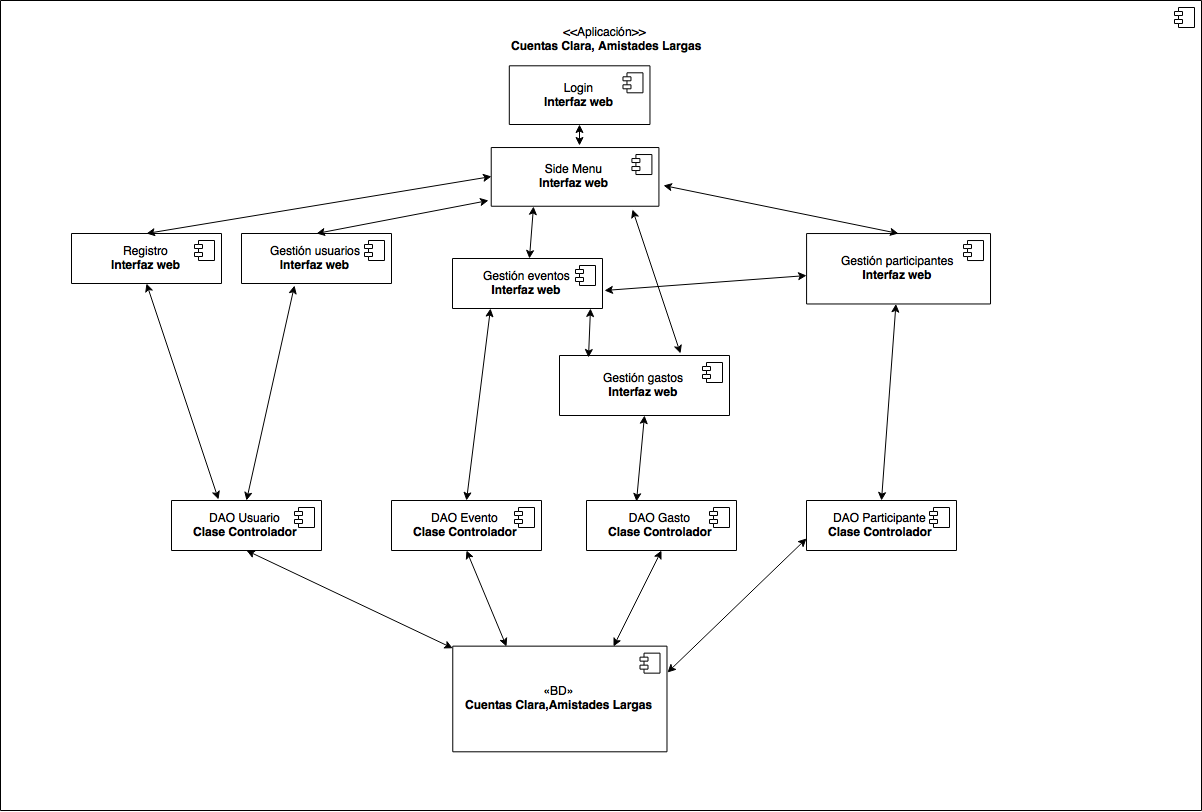
* Permitir identificar al usuario y permitir el acceso a la aplicación.
* Registrar un usuario nuevo para hacer uso de la aplicación.
* Gestionar los datos de los participantes a eventos, permitiendo crear, consultar, eliminar o actualizar los datos.
* Gestionar los datos de los eventos, permitiendo crear, consultar, eliminar o actualizar los datos.
* Añadir participantes y gastos al evento. También se podrá cerrar el evento y generar un informe con las cuentas.
* Realizar operaciones de administración.

Por otro lado, hay una serie de restricciones que no se contemplan en este proyecto:

* No es posible comunicarse con otros usuarios de la aplicación.
* La aplicación no permite realizar pagos entre los usuarios.
* No se puede dividir los gastos entre participante en partes desiguales.

# **Esquema funcional**

A continuación se muestra un esquema funcional de los componentes que forman el sistema de la aplicación.



El esquema se compone de 7 interfaces web, 4 clases controladoras y la base de datos.Las cuales se detallarán a continuación:

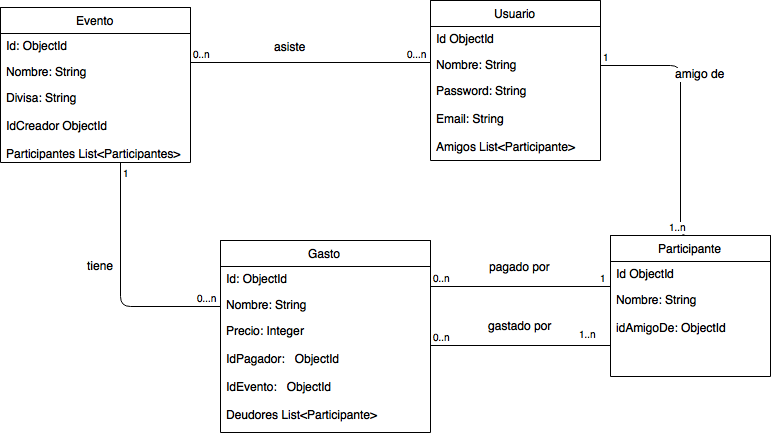
* Interfaz web ‘Login’: Interfaz web por defecto que muestra la aplicación y controla el acceso a la misma. La interfaz permite acceder a la interfaz ‘Registro’ en caso de que el usuario no esté dado de alta en el sistema.

En caso de que el usuario introduzca su usuario y contraseña correctamente accederá a la aplicación.

* Interfaz ‘Side Menú’: Interfaz web, mostrada como barra lateral que permite navegar entre las diferentes interfaces de la aplicación.
* Interfaz web ‘Registro’: Interfaz web que permite registrarse en la aplicación a un nuevo usuario. Una vez hecho el registro, vuelve al Login.
* Interfaz o conjunto de interfaces ‘Gestión usuarios’: encargada de realizar las operaciones de inserción, borrado, actualización y consulta de los usuarios. Esta herramienta será utilizada principalmente por el administrador.
* Interfaz o conjunto de interfaces ‘Gestión eventos: encargada de realizar las operaciones de inserción, borrado, actualización y consulta de los eventos.
* Interfaz o conjunto de interfaces ‘Gestión participantes: encargadas de añadir o eliminar participantes a un evento concreto.
* Interfaz o conjunto de interfaces ‘Gestión gastos: encargadas de añadir o eliminar gastos a un evento concreto.
* Controladores DAO (Usuarios, Evento, Gasto, Participante): Clases controladoras encargadas de recoger datos de las interfaces y enviarlas a la base de datos y viceversa.
* Base de datos: base de datos MongoDB encargada del albergar los datos de la aplicación.

# **Diagrama UML**

En el siguiente apartado se muestra un diagrama de las clases que forman la aplicación, junto con sus atributos y las relaciones entre ellas.



A continuación se detallarán las clases:

|  |  |  |  |
| --- | --- | --- | --- |
| **C-01: Evento** | | | |
| **Responsabilidades** | Clase responsable de operar con los datos de los eventos. Contendrá todos los métodos para insertar, borrar, consultar o actualizar, así como añadir gastos y participantes al evento. | | |
| **Atributos** | **Nombre** | **Tipo** | **Descripción** |
| id | ObjectId | Identificador del evento |
| nombre | String | Nombre del evento |
| divisa | String | Moneda utilizada en el evento |
| idCreador | ObjectId | Identificador del Usuario encargado de crear el evento. |
| participantes | List<Participante> | Listado de participantes del evento. |

|  |  |  |  |
| --- | --- | --- | --- |
| **C-02: Usuario** | | | |
| **Responsabilidades** | Clase responsable de operar con los datos de los usuarios. Contendrá todos los métodos para insertar, borrar, consultar o actualizar. | | |
| **Atributos** | **Nombre** | **Tipo** | **Descripción** |
| id | ObjectId | Identificador del usuario |
| nombre | String | Nombre del usuario |
| password | String | Contraseña del usuario |
| email | String | Correo del usuario |
| amigos | List<Participante> | Amigos creados por el usuario. |

|  |  |  |  |
| --- | --- | --- | --- |
| **C-03: Gasto** | | | |
| **Responsabilidades** | Clase responsable de operar con los datos de los gastos. Contendrá todos los métodos para insertar, borrar, consultar o actualizar. | | |
| **Atributos** | **Nombre** | **Tipo** | **Descripción** |
| id | ObjectId | Identificador del gasto |
| nombre | String | Nombre del gasto |
| precio | Integer | Cantidad del gasto. |
| idEvento | ObjectId | ID del evento al que pertenece el gasto. |
| idPagador | ObjectId | ID del pagador del evento. |
| Deudores | List<Participante> | Amigos participantes en el gasto. |

|  |  |  |  |
| --- | --- | --- | --- |
| **C-04: Participante** | | | |
| **Responsabilidades** | Clase responsable de operar con los datos de los participantes a eventos. Contendrá todos los métodos para insertar, borrar, consultar o actualizar. | | |
| **Atributos** | **Nombre** | **Tipo** | **Descripción** |
| id | ObjectId | Identificador del participante. |
| nombre | String | Nombre del participante |
| idAmigoDe | ObjectId | ID del usuario amigo del participante. |

# **Implementación**

En este apartado se detalla la estructura interna del proyecto. Describiendo la función de cada uno de sus componentes.

El proyecto está compuesto por varios paquetes:

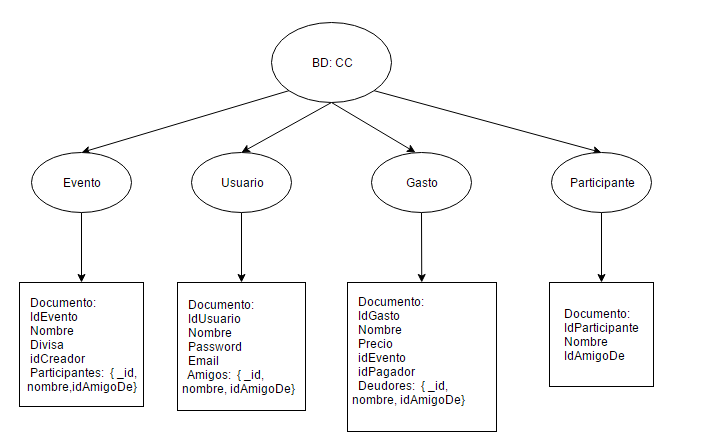
* **ccamistadeslargas/**
  + **AmigosLayout:** HorizontalSplitLayout encargado de gestionar las amistades del usuario. Se pueden crear, eliminar y actualizar.
  + **EventosLayout:** HorizontalSplitLayout encargado de gestionar los eventos creados del usuario. Se pueden crear, actualizar, eliminar, añadir participantes, añadir gastos y calcular cuentas de cada evento.
  + **IndexView:** Vista del SideMenu contenedor de los layout anteriores.
  + **LoginView:** Vista del login con la opción de logarse o registrarse.
  + **NavigatorUI:** UI encargada de relacionar todas las vistas.
  + **RegistrarView:** Vista del formulario de registro.
  + **Session:** Clase java encargada de controlar la sesión mediante un conjunto clave-valor.
* **ccamistadeslargas/modelo**
  + **Evento:** Clase que representa el evento.
  + **Gasto:** Clase que representa el gasto.
  + **Participante:** Clase que representa el participante.
  + **ResumenPlusvalia:** Clase encargada de guardar los datos del resumen de las cuentas que se mostrará en forma de tabla al finalizar un evento.
  + **Tupla:** Clase auxiliar de “ResumenPlusvalia” encargada de almacenar el deudor y el importe de un pago en forma de clave-valor.
  + **Usuario:** Clase que representa el usuario.
* **Ccamistadeslargas/DAO**
  + **EventoDAO:** DAO encargado de gestionar las operaciones con la BD relacionadas con los eventos.
  + **GastoDAO:** DAO encargado de gestionar las operaciones con la BD relacionadas con los gastos.
  + **ParticipanteDAO:** DAO encargado de gestionar las operaciones con la BD relacionadas con los participantes.
  + **UsuarioDAO:** DAO encargado de gestionar las operaciones con la BD relacionadas con los usuarios.

# **Esquema Maven: Diagrama de Dependencias**

# **Diagrama MongoDB**

En este apartado se detalla la estructura empleada a la hora de almacenar los datos en una base de datos en MongoDB.

La estructura se dividirá en cuatro documentos (evento,usuario,gasto,participante) los cuales tendrán una estructura definida como se detalla en la siguiente figura.



# **Manual de uso**

En el siguiente manual de uso, se identifican una serie de usuarios que pueden utilizar la aplicación. Dependiendo del tipo de usuario podrá realizar una operaciones determinadas u otras.

Se detallarán a continuación las operaciones por cada usuario.

## **Usuario sin registrar:**

El usuario sin identificar accederá a la pantalla de Login, que se trata de la página por defecto que muestra la aplicación.

[Insertar pantalla de login]

Esta pantalla de Login podremos realizar las operaciones de Login pero al tratarse de un usuario sin registrar, pincharemos en el botón ‘Sign Up’ para registrar al usuario.

Una vez pinchamos, la aplicación navegará a la pantalla Registrar como se muestra en la siguiente imagen.

[Insertar pantalla de registrarse]

En esta pantalla se nos muestra un formulario para darnos de alta en la aplicación. Rellenamos los datos solicitados (nombre, password y email) y pulsamos en ‘Sign Up’ dándonos de alta en la aplicación y accediendo a la misma.

## **Usuario Registrado:**

Por otro lado, si somos un usuario ya registrado en la aplicación. En la pantalla de Login nos bastará con introducir nuestro email y password.

Una vez hecho, accederemos a la pantalla principal de navegación.

[Insertar pantalla principal]

Esta pantalla se compone de una pantalla una menú lateral que nos permite navegar entre las diferentes pantallas de la aplicación.

Se compone de:

* Eventos
* Amigos
* Cerrar sesión

**1) Eventos**

En la sección Eventos se mostrará un listado de los eventos que tenga creados el usuario. Al seleccionar uno se mostrará los datos del mismo y una serie de botones.

[Insertar imagen de un evento]

Se muestra el nombre y la divisa del evento para que estos se puedan modificar. Más abajo se encuentra un listado de gastos asociados al evento y un listado de los participantes del mismo.

A esta información le acompaña una serie de botones:

* Actualizar evento: actualiza el valor nombre y divisa.
* Eliminar evento: elimina el evento.
* Añadir pago: se muestra un formulario para añadir un pago.

[insertar imagen formulario pago]

El precio indica la cantidad a pagar, el pagador es el participante que haya puesto el dinero y los deudores son los que participan en el pago.

* Añadir participante: muestra el listado de amistades creadas por el usuario y da la posibilidad de añadir uno al evento.
* Hacer las cuentas: recalcula las deudas de los pagos entre los participantes del evento.

[insertar imagen resumen deudas]

La primera tabla de este resumen informa de las deudas finales de cada participante a otro del grupo. A su vez muestra cuántos pagos han sido partícipes y cuántos han pagado.

La segunda tabla …

**2) Amigos**

En la sección Amigos se mostrará un listado de los amigos creados por el usuario. Estos serán futuros participantes de los eventos. Al seleccionar un amigo se podrá modificar su nombre.

[insertar imagen del listado y el formulario]

**3) Cerrar sesión**

Se cierra la sesión y vuelve a la vista del login.

## **Usuario Administrador:**

# **Extras**

1) Usar el Navigator de Vaadin para navegar por diferentes vistas de la aplicación.

2) Usar el componente de la extensión SideMenu de Vaadin [https://vaadin.com/directory#!addon/sidemenu-add-on]