



**Universidad
Europea** MADRID

PROYECTO INTEGRADOR

1 DAW

CEVICHE MADRILEÑO



Cristhian Andree Chafloque Chafloque

Hugo Rubio Crespo

Eduardo Alexander Utrilla Quispe

Índice

Resumen	2
1. Introducción	3
2. Objetivos	3
3. Tecnologías utilizadas	3
4. Desarrollo e implementación	3
5. Metodología	3
6. Resultados y conclusiones	4
7. Trabajos futuros	4
Anexos	5
Anexo I – Listado de requisitos de la aplicación	5
Anexo II – Guía de uso de la aplicación	5
Anexo III	5

Resumen

1. Introducción

El presente documento representa la memoria del Proyecto Integrador. En este, se unen los conocimientos y competencias adquiridos en tres asignaturas del curso: Programación, Bases de Datos y Entornos de Desarrollo. A través de su desarrollo, se pretende consolidar el aprendizaje práctico de los contenidos impartidos a lo largo del curso, al tiempo que se promueve el trabajo colaborativo y el uso de metodologías ágiles en un entorno simulado de desarrollo profesional.

El proyecto consiste en el diseño y la implementación de una aplicación de gestión para los espacios deportivos de la Universidad Europea de Madrid, atendiendo a una necesidad planteada por el polideportivo del centro. La aplicación permitirá gestionar salas, usuarios y actividades deportivas impartidas por el alumnado del ciclo de Actividad Física y Deportiva (TAFD), a través de una interfaz intuitiva con distintos roles de acceso y funcionalidades diferenciadas para usuarios y monitores.

Entre los principales objetivos del proyecto se encuentran: el desarrollo de una base de datos relacional normalizada, la implementación de una aplicación de escritorio en Java bajo el patrón de diseño Modelo-Vista-Controlador (MVC), y la gestión estructurada del trabajo en equipo mediante herramientas colaborativas como GitHub y Trello, siguiendo la metodología ágil SCRUM.

La aplicación desarrollada permitirá a los usuarios (alumnos de distintos ciclos) consultar las actividades disponibles, inscribirse o darse de baja, así como revisar sus actividades apuntadas. Por otro lado, los monitores podrán crear, modificar o eliminar actividades, siempre que se respeten ciertas condiciones como horarios válidos y disponibilidad de salas. Todo ello será gestionado mediante un sistema de login que distinguirá entre tipos de usuario, y que ofrecerá menús personalizados según el perfil.

Además de servir como producto funcional, este proyecto busca fomentar la responsabilidad individual y colectiva, la planificación del trabajo, la documentación técnica y la capacidad de adaptación a nuevos retos, emulando así las dinámicas de una empresa de desarrollo real. A lo largo de este documento, se detallarán los aspectos técnicos, metodológicos y organizativos llevados a cabo durante su realización.

2. Objetivos

A la hora de evaluar los objetivos relativos al Proyecto, es importante diferenciar entre los generales, aquellos que son propios del Proyecto y los que afectan a los desarrolladores del mismo.

Los generales se enfocan en saber utilizar la formación recibida durante el curso a nivel didáctico y enfocar los problemas de forma que la solución sea posible.

Los objetivos del Proyecto se basan en la parte técnica que hay que aplicar, dividida por materias para una mejor diferenciación.

En cuanto a las utilidades del programa, se plasman en los objetivos funcionales del Proyecto, que se basan en la funcionalidad que un cliente querría que la aplicación tuviera.

Por último, es importante señalar los riesgos y restricciones a los que nos enfrentamos los desarrolladores para poder alcanzar todos estos objetivos. Detallarlos es muy importante para poder prever un plan de acción que minimice sus efectos en todo lo posible.

Los objetivos :

1. Objetivos Generales:

- Gestionar el proyecto de forma que se adecúe a lo aprendido durante el curso.
- Saber solicitar ayuda a la persona indicada y respetar el trabajo y autonomía de los miembros del equipo.
- Tener un espíritu innovador que posibilite una mejora del Proyecto y un mejor ambiente.
- Tratar de forma eficaz y respetuosa los posibles conflictos que puedan surgir y darles una solución que contribuya al buen flujo del Proyecto y del equipo.

2. Objetivos Específicos:

- Bases de Datos:
 - Diseñar modelos lógicos normalizados derivados del diagrama entidad-relación.
 - Crear bases de datos en base al modelo relacional.
 - Realizar el diseño de la base de datos utilizando el lenguaje de definición de datos.
 - Realizar el diseño de la base de datos usando el lenguaje de definición de datos.
- Programación:
 - Realizar el diseño de clases para seguir el patrón MVC.
 - Implementar cada clase de forma que cada una cumpla con su objetivo.
 - Desarrollar una aplicación que pueda conectar con la información de la base de datos.
- Entornos de desarrollo:
 - Desarrollar el análisis y diseño de la aplicación mediante el uso de técnicas de ULM.
 - Documentar aplicaciones a través de la presente memoria.
 - Gestionar las diferentes versiones del Proyecto mediante el uso de Git colaborativo.
 - Realizar pruebas del software.
 - Seguir la metodología ágil SCRUM para el desarrollo y planificación en equipo del Proyecto.

3. Objetivos Funcionales:

- Permitir la gestión de los diferentes espacios del polideportivo.
- Administrar alumnos y monitores, incluyendo el registro, modificado y borrado de usuarios.
- Control de acceso mediante usuario y contraseña, distinguiendo a los dos tipos de usuarios.
- Dar rendimiento al sistema de forma que las peticiones se ejecuten de forma fluida.

4. Restricciones y riesgos:

- Presupuesto limitado a las capas gratuitas de los softwares de desarrollo.
- Posibles incompatibilidades con otros sistemas operativos distintos de Windows.
- Falta de experiencia del equipo de desarrollo con algunas tecnologías necesarias para el correcto funcionamiento del Proyecto.

3. Tecnologías utilizadas

Para el desarrollo del proyecto integrador se ha utilizado un conjunto de tecnologías que permiten cubrir tanto el desarrollo de la aplicación como su gestión, documentación y control de versiones. Estas herramientas han sido seleccionadas por su adecuación al entorno educativo, su compatibilidad con el lenguaje Java y su uso habitual en entornos profesionales reales.

- **Lenguaje de Programación:** Java

Se ha utilizado Java como lenguaje principal para la implementación de la lógica de negocio, interfaz gráfica y conexión con la base de datos. Es un lenguaje orientado a objetos y ampliamente utilizado en entornos corporativos.

- **Interfaz gráfica:** Java Swing / WindowBuilder

La interfaz de usuario se ha construido con Swing, un conjunto de herramientas gráficas de Java. Para facilitar el diseño visual de las ventanas, se ha empleado el plugin WindowBuilder en Eclipse, lo que permite una creación más ágil y visual de formularios.

- **Base de Datos:** MySQL

Se ha usado MySQL como sistema de gestión de bases de datos relacional. Se han diseñado las tablas, claves foráneas, índices y relaciones siguiendo un modelo normalizado. La conexión con Java se ha realizado mediante el conector JDBC.

- **IDE de Desarrollo:** Eclipse

Ha sido el entorno principal de programación utilizado para escribir y organizar el código Java, integrar bibliotecas, ejecutar el proyecto y gestionar los errores. Incluye herramientas para depuración, pruebas y documentación.

- **Herramientas de Modelado:** Draw.io

Para la elaboración de diagramas UML (casos de uso, clases, E/R, etc.) se ha empleado draw.io, que permite representar gráficamente la arquitectura del sistema.

- **Control de versiones y repositorio:** Git + GitHub

Se ha utilizado Git para el control de versiones del proyecto, y GitHub como plataforma colaborativa donde se aloja el repositorio, se documenta el código y se realiza el seguimiento del avance. Esto ha permitido a los miembros del equipo trabajar de manera sincronizada.

- **Gestión de tareas:** Trello

Para la organización del trabajo y seguimiento de tareas se ha utilizado Trello, aplicando un enfoque Kanban, con tarjetas organizadas en columnas según su estado: pendiente, en proceso, finalizado. Esto ha facilitado la planificación por sprints y la visualización del flujo de trabajo.

- **Pruebas:** JUnit

Se ha utilizado el framework JUnit para la creación de pruebas unitarias que validan el correcto funcionamiento de métodos y clases del proyecto.

- **Documentación:** JavaDoc y Markdown (GitHub Wiki)

La documentación técnica del código se ha generado con JavaDoc. Además, se ha elaborado un manual de usuario utilizando Markdown en la wiki del repositorio de GitHub, para facilitar su lectura en línea y su mantenimiento.

4. Metodología

Para el desarrollo del proyecto, se ha optado por utilizar una combinación de metodologías ágiles tales como SCRUM como marco principal de trabajo y Kanban para el seguimiento

visual de tareas a través de la herramienta Trello, además de un diagrama de Gantt que permita visualizar la planificación temporal de todo el proyecto.

Se ha aplicado SCRUM mediante la organización del desarrollo en Sprints por semanas, cada uno con entregables definidos. Esto permite una planificación iterativa e incremental, facilitando la adaptación a cambios y el control constante del avance.

En cada sprint se ha llevado a cabo:

- Una planificación inicial del sprint con definición de objetivos y tareas.
- Un Sprint Review al final, para presentar lo conseguido y reflexionar sobre mejoras .
- Una actualización del repositorio GitHub y del tablero Trello.
- Tareas asignadas al equipo usando Trello para la gestión visual de los avances.

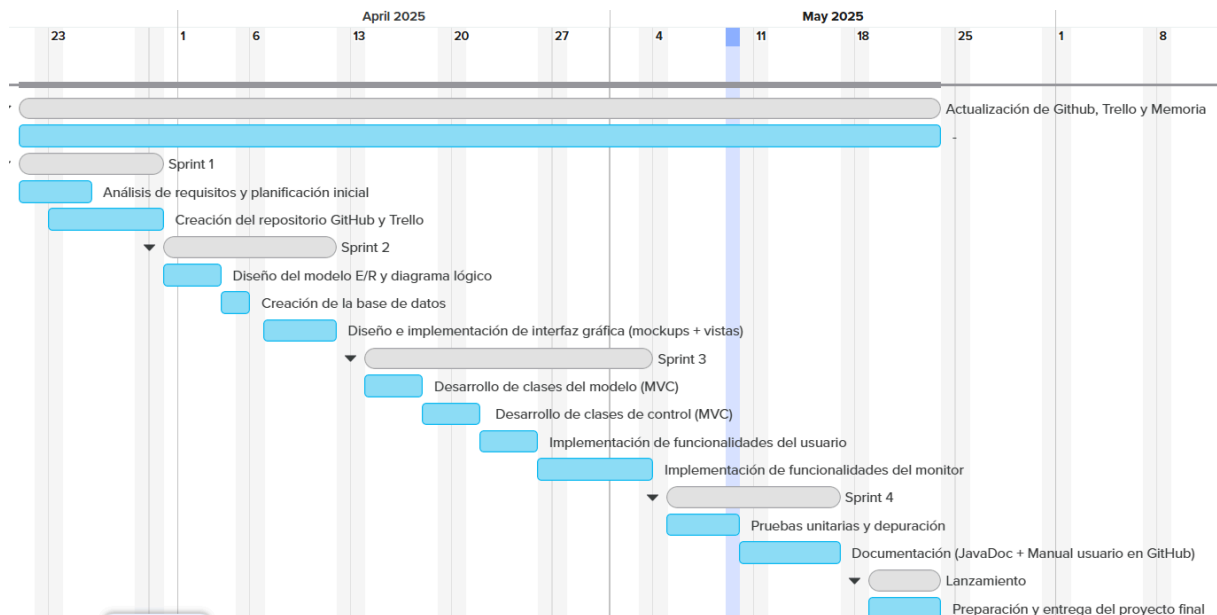
Cada miembro del equipo ha tenido un rol activo, organizando su carga de trabajo y aportando en las reuniones de sprint.

Además del enfoque SCRUM, se ha utilizado Kanban a través de Trello para representar gráficamente el estado de las tareas del proyecto, organizadas en columnas como: “Pendiente”, “En progreso”, “Hecho” y “Bloqueado”. Lo cual facilita la visualización del flujo de trabajo y permite gestionar mejor las prioridades del trabajo.



Complementariamente, se ha elaborado un diagrama de Gantt que permite observar la planificación global del proyecto, reflejando la temporalización de los sprints, los hitos importantes (revisiones, pruebas, entrega final) y la distribución del trabajo por módulos: Bases de Datos, Programación y Entornos de Desarrollo.

- Sprint 1 (21 de marzo – 28 marzo): Se llevó a cabo el análisis de los requisitos del proyecto y se desarrollaron los apartados de la memoria correspondiente a la entrega y se creó el repositorio en GitHub y la organización en Trello.
- Sprint 2 (31 de marzo – 11 de abril): Se diseñó el modelo de E/R y los diagramas lógicos para posteriormente crear la base de datos; también se diseñó e implementó la interfaz gráfica del programa en Java.
- Sprint 3 (21 de abril – 2 de mayo): Se desarrolló el diagrama de clases y posteriormente la creación de las clases pertenecientes al MVC.
- Sprint 4 (5 de mayo – 16 mayo): Se llevó a cabo pruebas en JUnit y se completó la documentación de la aplicación en JavaDoc así como también la realización del manual de usuario y se terminó la memoria.
- Lanzamiento: Elaboración de la presentación y revisión de la documentación.



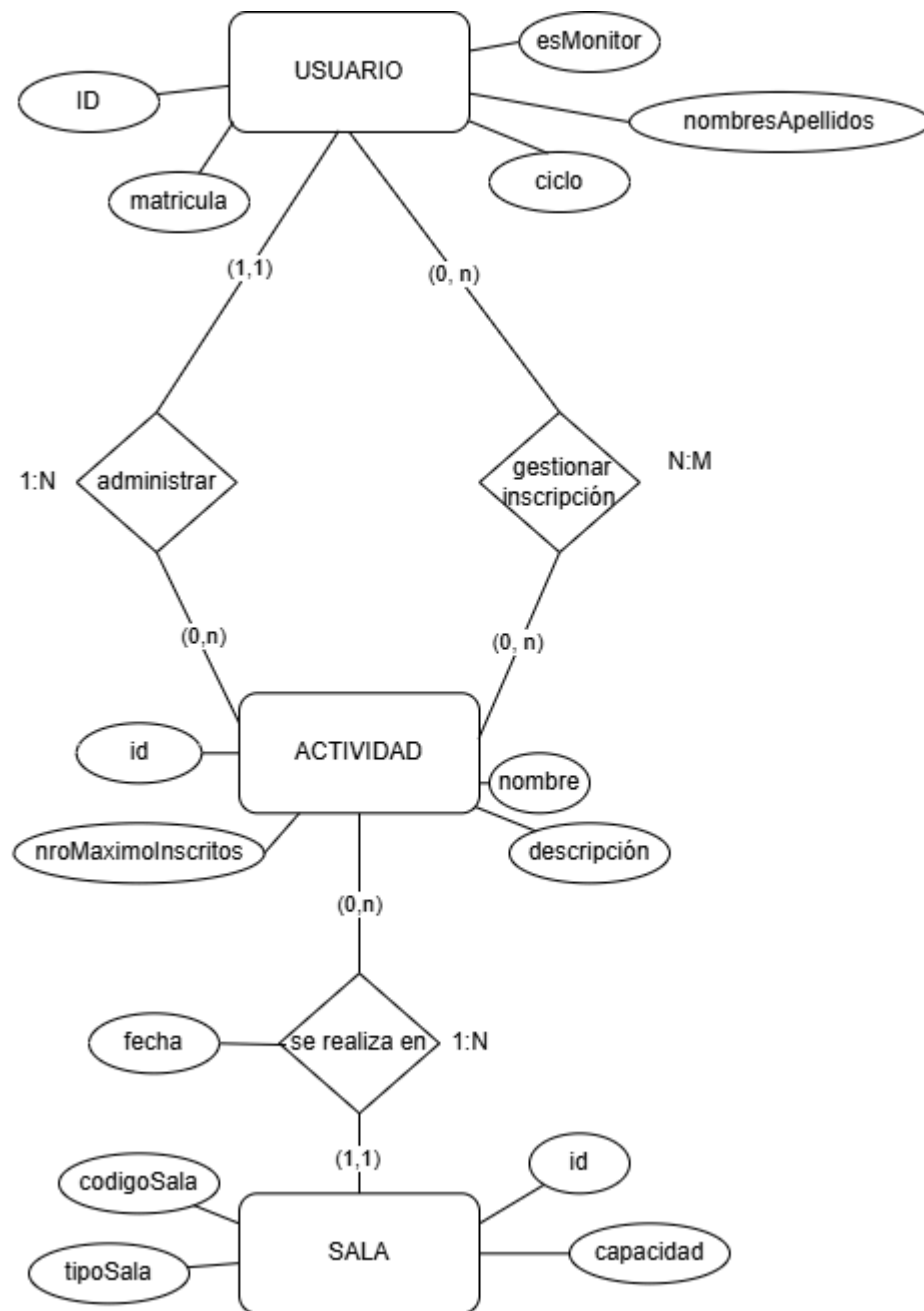
Link del Github: https://github.com/EduardoUQ/proyecto_integrador_cevichemadrileno

Link del Trello: <https://trello.com/b/rw4yhuy/proyecto-integrador>

5. Desarrollo e implementación

El Diagrama Entidad Relación mostrado a continuación se desarrolló en base a las entidades reconocidas para realizar las tablas de la base de datos junto con sus atributos y relaciones.

Se tienen reconocidas tres entidades principales que son Usuario, Actividad y Sala. Un usuario tiene un identificador único como clave primaria (PK), un número de matrícula, si es un monitor, el ciclo que cursa y sus nombres y apellidos; este puede gestionar su inscripción en la actividad inscrita si es un usuario normal, pero si es un monitor podrá también administrar. Una actividad tiene un identificador como clave primaria, un nombre, una descripción, el identificador del monitor que imparte la clase y el número máximo de inscritos. La actividad se realiza en una sala, cuya relación tiene una fecha en específico. La sala tiene un identificador como clave primaria, el tipo, un código y su capacidad.



Modelo Relacional:

Se crean las tablas con cada una de las 3 entidades. En cuanto a la relación N:M, se crea su correspondiente tabla con la clave primaria formada por la suma de las clave foráneas.

En la relaciones 1:N, dado que la cardinalidad mínima de la relación 1 es 1, se propaga la clave de esta tabla a la de la N.

USUARIO:(**id**, matrícula, esMonitor, nombreApellidos, ciclo)

PK: id

ACTIVIDAD:(**id**, id_monitor*, id_sala*, fecha, nombre, nroMaximoInscritos, descripción)

PK: id

FK: id_monitor->USUARIO(id)

SALA:(**id**, codigoSala, capacidad, tipoSala)

PK: id

INSCRIPCION:(**id_usuario***, **id_actividad***)

PK: id_usuario + id_actividad

FK:id_usuario ->USUARIO(id)

FK:id_actividad ->ACTIVIDAD(id)

Abordamos ahora la normalización. Las tablas se encuentran en primera forma normal (1FN) porque no existe posibilidad de valores multivaluados ni de tuplas duplicadas.

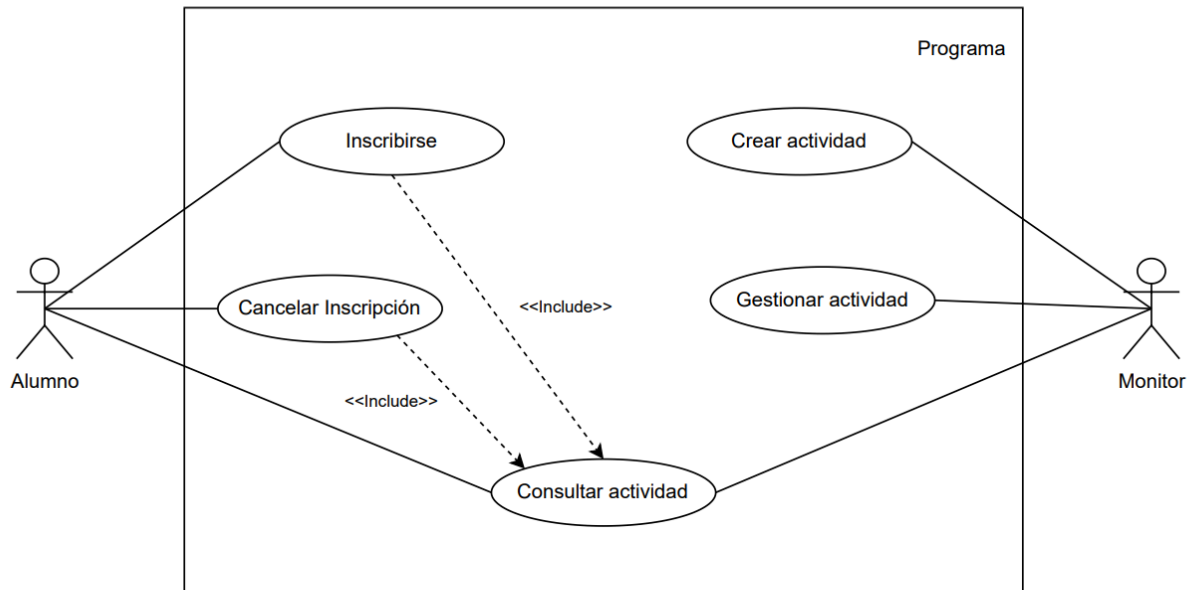
También está en 2FN porque todos los atributos secundarios tienen dependencia funcional total de la clave primaria.

La 3FN exige que no haya dependencias transitivas, de forma que los atributos secundarios dependan únicamente de la clave.

Por lo tanto se ha decidido que cada atributo de una tabla depende únicamente de su clave primaria, quedando como se muestra en la primera forma normal.

Diagrama de caso de uso:

En el diagrama de caso de uso se observa que consta de dos actores los cuales son el alumno y el monitor. Por parte del alumno, podrá inscribirse a una actividad, cancelar su inscripción y consultar la actividad; para poder inscribirse y cancelar primero tendrá que consultar la actividad cada que quiera realizar una de las dos funciones. El monitor podrá crear actividades y gestionar la actividad, este último incluye consultar y editar; también podrá realizar funciones como alumno que sería inscribirse a una actividad y cancelarla.



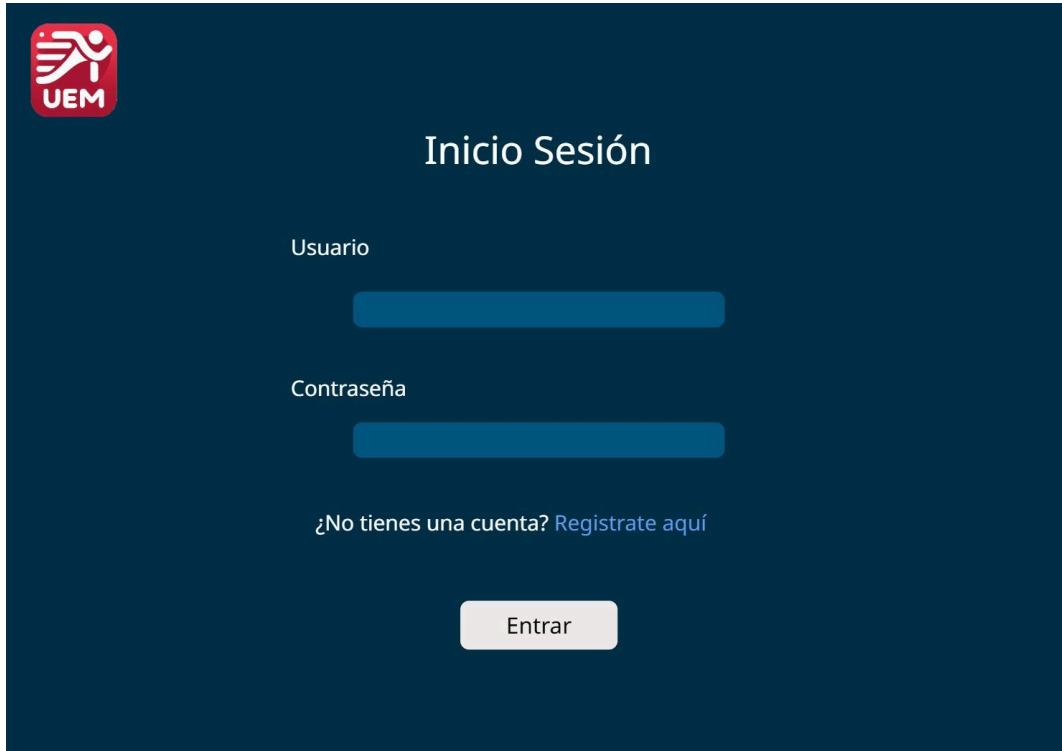
Logo de la aplicación UEM Move

El nombre de la aplicación está inspirado en el movimiento que se genera al realizar cualquier deporte u actividad, siendo una de las razones por la cual los alumnos se matriculan en las actividades que se dan en el polideportivo de la universidad; por lo que se busca incentivar más a los estudiantes a poder participar.


Respecto al logo, está representado por una figura humana en acción hecha con líneas simples representando una actividad física sin ser demasiado literal, dando la sensación de movimiento junto con el nombre de la universidad; así mismo los colores elegidos son los representativos rojo y blanco de la universidad.



Respecto a la interfaz del programa se tiene la siguiente vista principal que al iniciar la aplicación aparecerá un inicio de sesión para introducir el nombre de usuario y contraseña, también tendrá la opción de crearse una cuenta con los datos necesarios.



The login form is displayed on a dark blue background. In the top left corner is the UEM logo. The title 'Inicio Sesión' is centered at the top. Below it are two input fields: 'Usuario' and 'Contraseña'. A link '¿No tienes una cuenta? Regístrate aquí' is positioned below the password field. At the bottom center is a white 'Entrar' button.



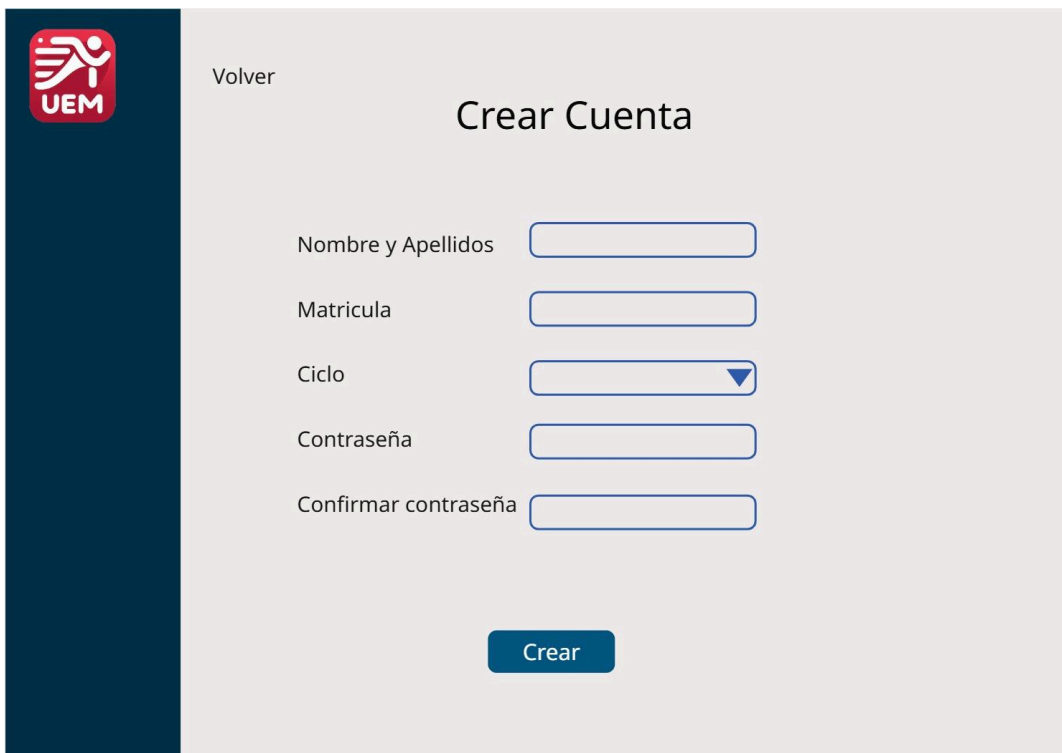
Inicio Sesión

Usuario


Contraseña

[¿No tienes una cuenta? Regístrate aquí](#)

Entrar



The create account form is shown on a light gray background with a dark blue sidebar on the left containing the UEM logo. A 'Volver' link is in the top left. The title 'Crear Cuenta' is centered. The form includes five input fields: 'Nombre y Apellidos', 'Matricula', 'Ciclo' (a dropdown menu), 'Contraseña', and 'Confirmar contraseña'. A dark blue 'Crear' button is at the bottom center.

 Volver

Crear Cuenta

Nombre y Apellidos

Matricula

Ciclo

Contraseña

Confirmar contraseña

Crear

Cada que se inicie sesión será a manera de alumno ya que para poder entrar como monitor se le entregará las credenciales y tendrá la vista como alumno pero tendrá además las opciones correspondientes a su usuario, esto con el fin de evitar que cualquiera se cree un usuario como monitor.

Ambos usuarios podrán inscribirse a una actividad y ver las actividad en la que se han inscrito y darse de baja si así lo desean. Por parte del monitor podrá crear y borrar actividades.

Vista principal de Usuario

Mis Actividades				
Actividades inscritas				
Yoga	Jueves	16:00-17:00	Sala Principal	⊗
Baile	Viernes	12:00-13:30	Salón de espejos	⊗

Vista principal de Monitor



- Mi perfil
- Mis actividades
- Actividades
- Crear actividad

Mis Actividades

Actividades inscritas

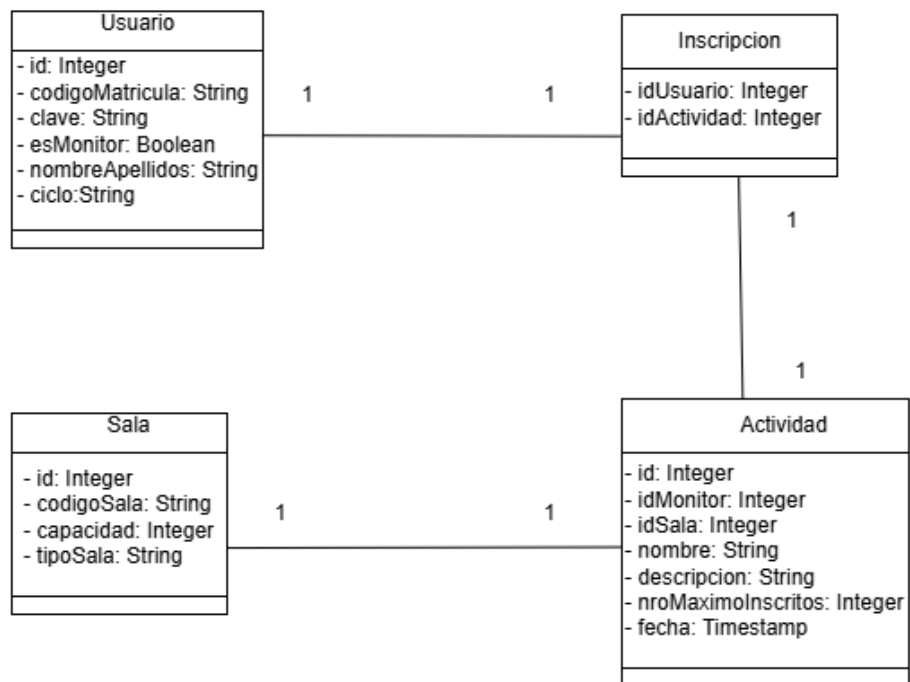
Yoga	Jueves	16:00-17:00	Sala Principal	
Baile	Viernes	12:00-13:30	Salón de espejos	

Actividades creadas

Futbol 5	Jueves	16:00-17:00	Cancha 2	Borrar
Baloncesto	Viernes	12:00-13:30	Gimnasio	Borrar

Diagramas de clase

En el diagrama de clase se puede observar que no existen relaciones de herencia, composición ni agregación entre las clases por lo que son todas las relaciones simples. Además, todos los atributos son privados (-) y ninguna clase tiene métodos propios (que no sean getters y setters). Finalmente, todas las cardinalidades son 1-1, al haber una clase llamada inscripción simplifica el sistema, ya que este diagrama representa una relación en la que la inscripción de un usuario responde únicamente a una actividad cuya realización se da también solo en una sala establecida en un único momento del día, sin poder repetirse.



6. Resultados y conclusiones

7. Trabajos futuros

Anexos

Anexo I – Listado de requisitos de la aplicación

Los requisitos de hardware y software para el desarrollo del proyecto son los siguientes:

1. Hardware:

- Requisitos mínimos:
 - CPU: Procesador de 2 núcleos
 - RAM: 4 GB
 - Almacenamiento: 500 MB
 - Conexión de red: 10 Mbps
 - Gráficos: Tarjeta integrada
- Requisitos recomendados:
 - CPU: Procesador de 4 núcleos
 - RAM: 8 GB
 - Almacenamiento: 500 MB
 - Conexión de red: 100 Mbps
 - Gráficos: Tarjeta integrada

2. Software:

- Entorno de desarrollo:
 - JDK de Java
 - Eclipse IDE
- Diseño:
 - Software Ideas Modeler o draw.io
- Bases de Datos:
 - MySQL u Oracle SQL
- Control de versiones:
 - Git
- Planificación y trabajo en equipo:
 - Trello
 - Microsoft Office
 - Whatsapp

En cuanto a los requisitos de la aplicación, se recomienda al usuario los siguientes:

- Hardware:
 - CPU: Procesador de 2 núcleos
 - RAM: 4 GB
 - Almacenamiento: 500 MB
 - Gráficos: Tarjeta integrada
- Software:
 - Sistema Operativo: Windows, Linux o MacOS.
 - Java JRE
 - MySQL Server

Anexo II – Guía de uso de la aplicación

Anexo III - Historial de convivencia

En el Sprint 1, el equipo se organizó para poder realizar el avance correspondiente a las tareas de ese sprint, se repartió el trabajo de manera equitativa. Para una mejor coordinación se creó un grupo de whatsapp con los integrantes, así como también un link de jitsi para las reuniones continuas.

En el Sprint 2, se desarrollaron los apartados correspondientes a la parte de programación, así como también el diseño de vistas , logo y creación de los diagramas. Se tuvo que corregir algunos puntos por el feedback obtenido de la corrección del Sprint 1 correspondiente al diagrama entidad relación para la posterior creación de las tablas y clases del proyecto.

En el Sprint 3, se realizó la creación de clases correspondientes en Java con el patrón MVC. Uno de los integrantes (Hugo) se tuvo que ir de prácticas, no obstante, se siguió manteniendo la comunicación para el desarrollo del proyecto cumpliendo con su parte actualizando algunos apartados.