**Historias de Usuario**

**1. Historia de Usuario: Registro de Usuarios**

**Título: Registro de nuevos usuarios (Estudiantes y Profesores)**

**Como un nuevo usuario tendrá la capacidad de registrarse| en el sistema seleccionando mi rol (Estudiante o Profesor),**

**Para que pueda acceder a las funcionalidades específicas de mi rol dentro de la plataforma.**

**Criterios de Aceptación:**

* **El usuario debe proporcionar un nombre de usuario único, contraseña y nombre completo.**
* **El sistema debe permitir seleccionar entre los roles de Estudiante y Profesor durante el registro.**
* **Si el nombre de usuario ya existe, el sistema debe notificar al usuario y solicitar otro nombre de usuario.**
* **Las contraseñas deben coincidir durante la confirmación.**

**2. Historia de Usuario: Inicio de Sesión**

**Título: Autenticación de usuarios**

**Como un usuario registrado,**

**Quiero iniciar sesión en el sistema proporcionando mi nombre de usuario y contraseña,**

**Para que pueda acceder a mis Learning Paths y otras funcionalidades según mi rol.**

**Criterios de Aceptación:**

* **El usuario debe ingresar un nombre de usuario y contraseña válidos.**
* **Si las credenciales son incorrectas, el sistema debe notificar al usuario.**
* **Al iniciar sesión correctamente, el sistema debe redirigir al usuario al menú correspondiente a su rol.**

**3. Historia de Usuario: Creación de Learning Paths**

**Título: Crear nuevos Learning Paths**

**Como un profesor,**

**Quiero crear nuevos Learning Paths agregando actividades como Revisiones de Recursos, Tareas, Quizzes, Exámenes de Preguntas Abiertas y Encuestas,**

**Para que pueda estructurar el contenido educativo que los estudiantes seguirán.**

**Criterios de Aceptación:**

* **El profesor debe proporcionar detalles como título, descripción, objetivos y nivel de dificultad al crear un Learning Path.**
* **El profesor puede agregar múltiples actividades de diferentes tipos al Learning Path.**
* **El sistema debe asignar una versión inicial al Learning Path y permitir actualizaciones posteriores.**
* **Los Learning Paths creados deben guardarse persistente y correctamente asociado al profesor creador.**

**4. Historia de Usuario: Visualización de Learning Paths**

**Título: Ver Learning Paths creados por el profesor**

**Como un profesor,**

**Quiero ver una lista de todos los Learning Paths que he creado,**

**Para que pueda gestionarlos y monitorear la participación de los estudiantes.**

**Criterios de Aceptación:**

* **El profesor puede acceder a una lista de sus Learning Paths.**
* **Cada Learning Path muestra información relevante como título, descripción, objetivos, nivel de dificultad, duración total y versión.**
* **Si el profesor no ha creado ningún Learning Path, el sistema debe notificarlo.**

**5. Historia de Usuario: Inscripción en Learning Paths**

**Título: Inscribirse en Learning Paths disponibles**

**Como un estudiante,**

**Quiero ver y seleccionar entre los Learning Paths disponibles para inscribirme,**

**Para que pueda comenzar a seguir el contenido educativo proporcionado.**

**Criterios de Aceptación:**

* **El estudiante puede ver una lista de Learning Paths disponibles que no ha inscrito previamente.**
* **El sistema muestra detalles de cada Learning Path, incluyendo el creador.**
* **El estudiante puede seleccionar un Learning Path para inscribirse.**
* **Una vez inscrito, el Learning Path aparece en la lista de Learning Paths del estudiante.**

**6. Historia de Usuario: Gestión de Progreso en Learning Paths**

**Título: Gestionar y monitorear el progreso en Learning Paths**

**Como un estudiante,**

**Quiero ver y gestionar mi progreso en los Learning Paths inscritos,**

**Para que pueda realizar actividades y seguir mi avance.**

**Criterios de Aceptación:**

* **El estudiante puede ver una lista de sus Learning Paths inscritos con el porcentaje de completitud.**
* **Al seleccionar un Learning Path, el estudiante puede ver las actividades pendientes y completadas.**
* **El estudiante puede interactuar con cada actividad según su tipo (por ejemplo, responder encuestas, entregar tareas).**
* **El sistema actualiza el estado de las actividades y el porcentaje de completitud en tiempo real.**

**7. Historia de Usuario: Visualización de Estudiantes Inscritos**

**Título: Ver estudiantes inscritos en los Learning Paths del profesor**

**Como un profesor,**

**Quiero ver una lista de todos los estudiantes inscritos en cada uno de mis Learning Paths,**

**Para que pueda monitorear la participación y el progreso de los estudiantes.**

**Criterios de Aceptación:**

* **El profesor puede acceder a una lista de sus Learning Paths.**
* **Para cada Learning Path, el profesor puede ver los estudiantes inscritos con detalles como nombre y nombre de usuario.**
* **Si no hay estudiantes inscritos en un Learning Path específico, el sistema debe notificarlo.**

**8. Historia de Usuario: Revisión de Respuestas a Encuestas y Exámenes**

**Título: Revisar respuestas a encuestas y exámenes de estudiantes**

**Como un profesor,**

**Quiero acceder y revisar las respuestas que los estudiantes han proporcionado a encuestas y exámenes de preguntas abiertas,**

**Para que pueda evaluar y proporcionar retroalimentación adecuada.**

**Criterios de Aceptación:**

* **El profesor puede seleccionar un Learning Path y acceder a las respuestas de encuestas y exámenes asociadas.**
* **Las respuestas se muestran de manera organizada, agrupadas por estudiante.**
* **El sistema debe permitir al profesor agregar comentarios o calificaciones si es necesario.**

**Restricciones**

1. **Unicidad de Nombre de Usuario:**
   * **Cada nombre de usuario debe ser único en el sistema para evitar conflictos y asegurar una identificación correcta de los usuarios.**
2. **Roles Definidos:**
   * **Solo se permiten dos roles de usuario: Estudiante y Profesor. No se pueden asignar otros roles sin modificar el sistema.**
3. **Acceso Basado en Roles:**
   * **Las funcionalidades disponibles están restringidas según el rol del usuario. Por ejemplo, solo los profesores pueden crear Learning Paths, mientras que solo los estudiantes pueden inscribirse en ellos.**
4. **Persistencia de Datos:**
   * **Todos los datos deben ser persistentes y almacenados correctamente para garantizar que la información no se pierda entre sesiones.**
5. **Validación de Entradas:**
   * **Todas las entradas proporcionadas por los usuarios deben ser validadas para asegurar que cumplan con los formatos y restricciones establecidos (por ejemplo, niveles de dificultad entre 1 y 5).**
6. **Seguridad de Contraseñas:**
   * **Las contraseñas deben almacenarse de manera segura (por ejemplo, mediante hash) para proteger la información sensible de los usuarios.**

**Reglas de Dominio del Problema**

1. **Asociación de Usuarios y Learning Paths:**
   * **Cada Learning Path está asociado a un único profesor creador.**
   * **Un Estudiante puede estar inscrito en múltiples Learning Paths, y cada Learning Path puede tener múltiples Estudiantes inscritos.**
2. **Tipos de Actividades:**
   * **Las actividades dentro de un Learning Path pueden ser de diferentes tipos: Revisión de Recursos, Tareas, Quizzes, Exámenes de Preguntas Abiertas, y Encuestas.**
   * **Cada tipo de actividad tiene comportamientos y requisitos específicos (por ejemplo, los Quizzes tienen preguntas de opción múltiple con respuestas correctas predefinidas).**
3. **Progreso y Estados de Actividades:**
   * **Cada Estudiante tiene un progreso individual en cada Learning Path inscrito.**
   * **Las actividades pueden tener estados como Pendiente, Completada, Entregada, o Fallida, dependiendo de la interacción del estudiante.**
4. **Versionado de Learning Paths:**
   * **Cada vez que un profesor actualiza un Learning Path, la versión se incrementa para mantener un historial de cambios.**
5. **Evaluación y Retroalimentación:**
   * **Los profesores pueden evaluar las respuestas de los estudiantes a encuestas y exámenes, proporcionando retroalimentación y calificaciones según sea necesario.**

**Requerimientos Funcionales**

1. **Gestión de Usuarios:**
   * **Registro de nuevos usuarios con selección de rol.**
   * **Autenticación de usuarios mediante nombre de usuario y contraseña.**
   * **Gestión de perfiles de usuario.**
2. **Creación y Gestión de Learning Paths:**
   * **Creación de nuevos Learning Paths por parte de profesores.**
   * **Agregar, editar y eliminar actividades dentro de un Learning Path.**
   * **Versionado automático de Learning Paths al realizar actualizaciones.**
3. **Inscripción y Gestión de Progreso:**
   * **Inscripción de estudiantes en Learning Paths disponibles.**
   * **Visualización del progreso del estudiante en cada Learning Path.**
   * **Interacción con actividades específicas según su tipo.**
   * **Actualización automática de estados de actividades y cálculo de porcentaje de completitud.**
4. **Monitoreo y Evaluación por Profesores:**
   * **Visualización de estudiantes inscritos en los Learning Paths creados.**
   * **Acceso y revisión de respuestas a encuestas y exámenes de preguntas abiertas.**
   * **Provisión de retroalimentación y calificaciones a los estudiantes.**
5. **Persistencia y Seguridad de Datos:**
   * **Almacenamiento persistente de todos los datos relevantes (usuarios, Learning Paths, progresos).**
   * **Seguridad en el almacenamiento y manejo de contraseñas.**
6. **Interfaz de Usuario Intuitiva:**
   * **Menús claros y opciones accesibles según el rol del usuario.**
   * **Mensajes de error y validación de entradas amigables y comprensibles.**

**Requerimientos No Funcionales**

1. **Usabilidad:**
   * **La interfaz de consola debe ser fácil de navegar y entender para usuarios con diferentes niveles de experiencia técnica.**
2. **Escalabilidad:**
   * **El sistema debe ser capaz de manejar un creciente número de usuarios y Learning Paths sin degradar su rendimiento.**
3. **Seguridad:**
   * **Protección de datos sensibles mediante técnicas de encriptación y buenas prácticas de seguridad.**
   * **Prevención de accesos no autorizados mediante autenticación robusta.**
4. **Mantenibilidad:**
   * **Código modular y bien documentado para facilitar futuras modificaciones y mejoras.**
   * **Implementación de pruebas unitarias para asegurar la calidad del código.**
5. **Rendimiento:**
   * **Respuesta rápida a las acciones del usuario, incluso cuando se manejan grandes volúmenes de datos.**
6. **Compatibilidad:**
   * **El sistema debe ser compatible con diferentes sistemas operativos que soporten Java.**
7. **Portabilidad:**
   * **Facilitar la migración del sistema a diferentes entornos sin requerir cambios significativos en el código.**
8. **Fiabilidad:**
   * **El sistema debe ser resistente a fallos, asegurando la integridad de los datos y minimizando el tiempo de inactividad.**
9. **Documentación:**
   * **Proporcionar documentación clara y detallada para usuarios finales y desarrolladores.**

**Explicacion  
  
1. Separacion de las Responsabilidades: (Principio de Responsabilidad Unica y Exclusiva)  
Para facilitar la división de las tareas que debía cumplir cada parte dividimos el trabajo en distintas clases, a pesar de que debido a la falta de tiempo y organización algunas clases que aun están presentes en el proyecto no son empleadas ni están conectadas, procuramos que los paquetes especificos se mantuvieran**

** users: Gestiona todas las clases relacionadas con los usuarios del sistema (User, Teacher, Student).**

** activities: Contiene las diferentes clases de actividades (Assignment, Quiz, etc.) y sus componentes asociados (Question, SurveyQuestion).**

** data: Encapsula la lógica de persistencia de datos a través de la clase DataManager.**

** interfaces: Define contratos como IActivityCreator que facilitan la extensibilidad y flexibilidad del sistema.**

** main: Incluye la clase principal y la interfaz de usuario (ConsoleInterface).**

**Este orden permite que el mantenimiento, legibilidad y escalabilidad del proyecto sea mas sencilla al permitir que cada clase y paquete se modifique de manera independiente**

**2. Uso de Clases Abstractas e Interfaces**

* **Clases Abstractas (User, Activity):**
  + **User: Define atributos y métodos comunes para todos los usuarios, asegurando que las subclases (Teacher, Student) implementen comportamientos específicos mediante métodos abstractos como getRole().**
  + **Activity: Establece una base para diferentes tipos de actividades, obligando a las subclases a implementar métodos como getType() y copy(), lo que garantiza una interfaz consistente.**

**Interfaces (IActivityCreator):**

* **Permiten definir contratos claros para la creación de actividades, facilitando múltiples implementaciones (por ejemplo, ConsoleInterface para consola y potencialmente GUIActivityCreator para una interfaz gráfica).**

**3. Herencia y Polimorfismo**

* **Herencia:**
  + **Teacher y Student heredan de User, reutilizando atributos y métodos comunes mientras añaden funcionalidades específicas.**
  + **Tipos de Activity heredan de la clase abstracta Activity, promoviendo la reutilización de código y la consistencia en la manipulación de diferentes actividades.**
* **Polimorfismo:**
  + **Permite tratar diferentes tipos de actividades de manera uniforme a través de referencias de tipo Activity, simplificando la gestión y ejecución de actividades diversas.**

**4. Encapsulación y Abstracción**

* **Encapsulación:**
  + **Los atributos de las clases están generalmente declarados como private o protected, accedidos mediante métodos getter y setter, protegiendo el estado interno de las clases.**
* **Abstracción:**
  + **Las clases abstractas e interfaces ocultan la complejidad interna y exponen solo lo necesario, facilitando el uso y la interacción entre componentes sin necesidad de conocer detalles de implementación.**

**5. Persistencia y Serialización**

* **Serialización (Serializable):**
  + **Todas las clases que necesitan ser persistidas implementan Serializable, permitiendo que los objetos se guarden y recuperen fácilmente desde archivos, garantizando la persistencia de datos entre sesiones de la aplicación.**
* **DataManager:**
  + **Centraliza la lógica de carga y guardado de datos, promoviendo la reutilización y evitando la duplicación de código en diferentes partes del sistema.**

**6. Flexibilidad y Extensibilidad**

* **Interfaces y Clases Modulares:**
  + **La utilización de interfaces como IActivityCreator y la estructuración modular permiten agregar nuevas funcionalidades (por ejemplo, nuevos tipos de actividades o diferentes interfaces de usuario) sin afectar significativamente el resto del sistema.**
* **Métodos de Copia y Constructores de Copia:**
  + **Facilitan la creación de instancias duplicadas de objetos (LearningPath), permitiendo funcionalidades como copiar y asignar Learning Paths a diferentes profesores de manera eficiente y consistente.**

**7. Interfaz de Usuario Centralizada**

* **ConsoleInterface:**
  + **Actúa como el punto central de interacción con el usuario, gestionando el flujo de la aplicación y delegando responsabilidades específicas a través de métodos bien definidos.**
* **Implementación de IActivityCreator:**
  + **Asegura que la creación de actividades sea consistente y estandarizada, permitiendo que ConsoleInterface maneje diferentes tipos de actividades de manera uniforme.**

**8. Mantenibilidad y Escalabilidad**

* **Código Organizado y Documentado:**
  + **La estructura clara y la documentación detallada facilitan la comprensión del sistema, permitiendo que nuevos desarrolladores se integren fácilmente y que se realicen modificaciones o ampliaciones con menor riesgo de introducir errores.**
* **Modularidad:**
  + **La capacidad de aislar y actualizar componentes individuales sin afectar el sistema completo mejora la capacidad de adaptación del proyecto a nuevas necesidades o cambios en los requisitos.**

** Modelo (src.com.learningpath)**

* **Incluye clases como LearningPath, Activity, Progress, y subclases de Activity (Assignment, ResourceReview, Quiz, etc.), que definen los objetos centrales del sistema.**
* **La clase LearningPath representa un camino de aprendizaje con actividades asociadas.**
* **Progress rastrea el avance de los estudiantes en cada camino de aprendizaje.**

** Usuarios (src.com.learningpath.users)**

* **Define las clases User, Student, y Teacher.**
* **Incluye autenticación y roles de usuario para estudiantes y profesores.**

** Persistencia (src.com.learningpath.data)**

* **DataManager maneja la carga y el guardado de datos persistentes para LearningPath, Progress, y usuarios, utilizando archivos serializados.**

** Pruebas (src.com.learningpath.test)**

* **Contiene pruebas unitarias para las funcionalidades críticas de las clases principales.**
* **Pruebas de integración para validar historias de usuario y la interacción entre componentes.**

**Funcionamiento de la Persistencia**

**La persistencia de datos se implementa mediante la clase DataManager:**

* **Guardado de Datos: Utiliza ObjectOutputStream para serializar listas de objetos (LearningPath, Progress, y usuarios) y guardarlas en archivos .dat.**
* **Carga de Datos: Utiliza ObjectInputStream para deserializar los objetos guardados y restaurarlos en memoria.**
* **Pruebas de Persistencia: Los tests verifican que los datos guardados y cargados sean consistentes (ej., los títulos de los LearningPath son los mismos antes y después de la persistencia).**

##### **Pruebas Unitarias**

1. **LearningPathTest**
   * **Verifica que las actividades se agreguen correctamente.**
   * **Asegura que la duración total del camino se recalcula al agregar actividades.**
   * **Comprueba que los atributos se inicialicen correctamente en el constructor.**
2. **ProgressTest**
   * **Calcula el porcentaje de actividades completadas, considerando solo las obligatorias.**
   * **Asegura que el estado de las actividades se actualice correctamente (PENDING, COMPLETED, etc.).**
3. **DataManagerTest**
   * **Valida el guardado y carga de LearningPath.**
   * **Usa un directorio temporal para asegurar que las pruebas sean independientes del sistema.**
4. **UserRegistrationAndAuthenticationTest**
   * **Prueba la creación de usuarios (Teacher, Student) y su autenticación.**
   * **Valida que las credenciales correctas e incorrectas se manejen adecuadamente.**

##### **Pruebas de Integración**

* **Historias de Usuario Cubiertas:**
  1. **Un profesor puede crear un camino de aprendizaje con actividades y guardar los datos.**
  2. **Un estudiante puede inscribirse en un camino, completar actividades y rastrear su progreso.**
  3. **Un usuario puede autenticarse correctamente en el sistema.**

### ****Requisitos Funcionales y su Relación con las Pruebas****

#### **1. Creación y Administración de Caminos de Aprendizaje (Learning Paths)**

**Requisitos:**

* **Los profesores deben poder crear caminos de aprendizaje con múltiples actividades.**
* **Los caminos de aprendizaje deben tener atributos como título, descripción, objetivos y actividades.**

**Pruebas Relevantes:**

* **LearningPathTest**
  + **Prueba testAddActivityAndRecalculateDuration: Verifica que las actividades se agregan correctamente al camino de aprendizaje y que la duración total se recalcula.**

**java**

**Copiar código**

**assertEquals(2, lp.getActivities().size());**

**assertEquals(90, lp.getDuration());**

* + **Prueba testConstructorInitializesFieldsCorrectly: Confirma que los atributos como título, creador, fecha de creación y versión inicial se configuran correctamente.**

**java**

**Copiar código**

**assertEquals("Advanced Java", lp.getTitle());**

**assertNotNull(lp.getCreationDate());**

**assertEquals("1.0", lp.getVersion());**

**Cobertura: Estas pruebas aseguran que los caminos de aprendizaje se crean con los atributos requeridos y que su administración básica (agregar actividades) funciona como se espera.**

#### **2. Progreso de Estudiantes en Caminos de Aprendizaje**

**Requisitos:**

* **Los estudiantes deben poder inscribirse en un camino de aprendizaje.**
* **El sistema debe rastrear el estado de las actividades (pendiente, completado, etc.).**
* **Debe calcularse el porcentaje de completitud de un camino.**

**Pruebas Relevantes:**

* **ProgressTest**
  + **Prueba testCalculateCompletionPercentage: Valida que el porcentaje de completitud se calcule correctamente, considerando solo las actividades obligatorias.**

**java**

**Copiar código**

**assertEquals(50.0, progress.calculateCompletionPercentage());**

**progress.updateActivityStatus(activity2, ActivityStatus.COMPLETED);**

**assertEquals(100.0, progress.calculateCompletionPercentage());**

* + **Prueba testUpdateActivityStatus: Confirma que el estado de una actividad se actualiza correctamente.**

**java**

**Copiar código**

**progress.updateActivityStatus(activity, ActivityStatus.COMPLETED);**

**assertEquals(ActivityStatus.COMPLETED, progress.getActivityStatuses().get(activity));**

**Cobertura: Estas pruebas aseguran que el sistema rastree correctamente el progreso de los estudiantes en sus caminos de aprendizaje, incluyendo el estado de actividades y el cálculo de completitud.**

#### **3. Persistencia de Datos**

**Requisitos:**

* **Los datos de usuarios, caminos de aprendizaje y progreso deben persistir entre sesiones.**
* **El sistema debe guardar y cargar correctamente los datos.**

**Pruebas Relevantes:**

* **DataManagerTest**
  + **Prueba testSaveAndLoadLearningPaths: Valida que los caminos de aprendizaje se guarden y carguen correctamente desde archivos.**

**java**

**Copiar código**

**assertEquals("LP1", loadedLearningPaths.get(0).getTitle());**

**assertEquals(2, loadedLearningPaths.size());**

**Cobertura: Esta prueba garantiza que la persistencia funcione correctamente, manteniendo la integridad de los datos entre sesiones.**

#### **4. Autenticación y Gestión de Usuarios**

**Requisitos:**

* **Los usuarios deben poder registrarse como estudiantes o profesores.**
* **Los usuarios deben poder autenticarse con credenciales válidas.**

**Pruebas Relevantes:**

* **UserRegistrationAndAuthenticationTest**
  + **Prueba testUserRegistrationAndAuthentication: Valida que los usuarios se registren correctamente y puedan autenticarse.**

**java**

**Copiar código**

**assertTrue(foundTeacher.authenticate("securePass"));**

**assertFalse(foundTeacher.authenticate("wrongPass"));**

**Cobertura: Esta prueba asegura que el sistema maneje correctamente la creación y autenticación de usuarios, garantizando seguridad y control de acceso.**

#### **5. Casos de Uso Complejos (Pruebas de Integración)**

**Requisitos:**

* **Los profesores deben poder crear y administrar caminos de aprendizaje.**
* **Los estudiantes deben poder inscribirse en caminos, completar actividades y rastrear su progreso.**

**Pruebas Relevantes:**

* **Pruebas de Integración:**
  + **testStudentEnrollmentAndProgress: Valida el flujo completo de inscripción y progreso de un estudiante en un camino de aprendizaje.**

**java**

**Copiar código**

**assertEquals(0.0, progress.calculateCompletionPercentage());**

**progress.updateActivityStatus(activity1, ActivityStatus.COMPLETED);**

**assertEquals(100.0, progress.calculateCompletionPercentage());**

**Cobertura: Las pruebas de integración validan casos de uso completos, asegurando que todas las piezas del sistema (creación de caminos, inscripción, progreso, persistencia) trabajen juntas correctamente.**