DIAGRAMA A.

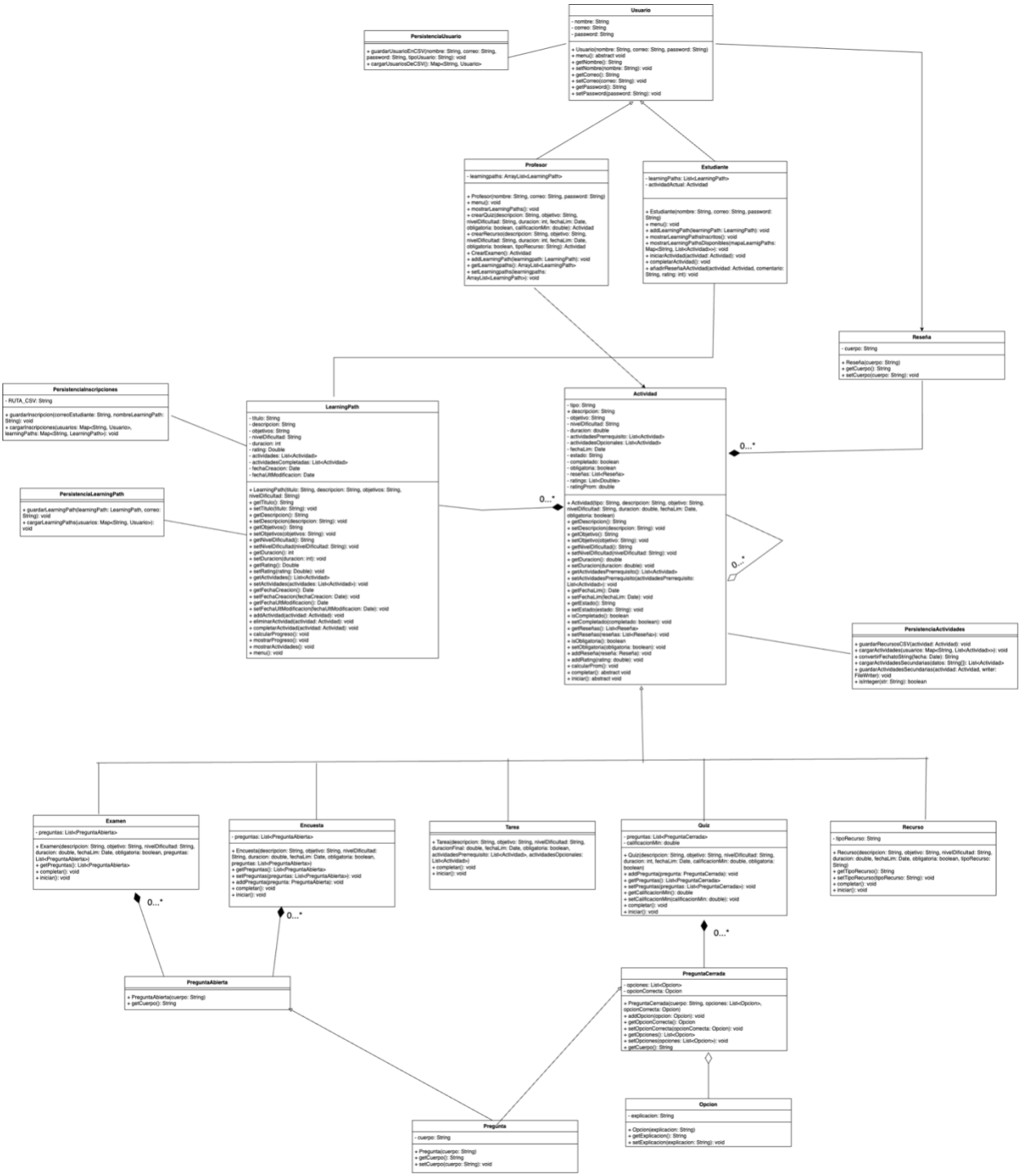
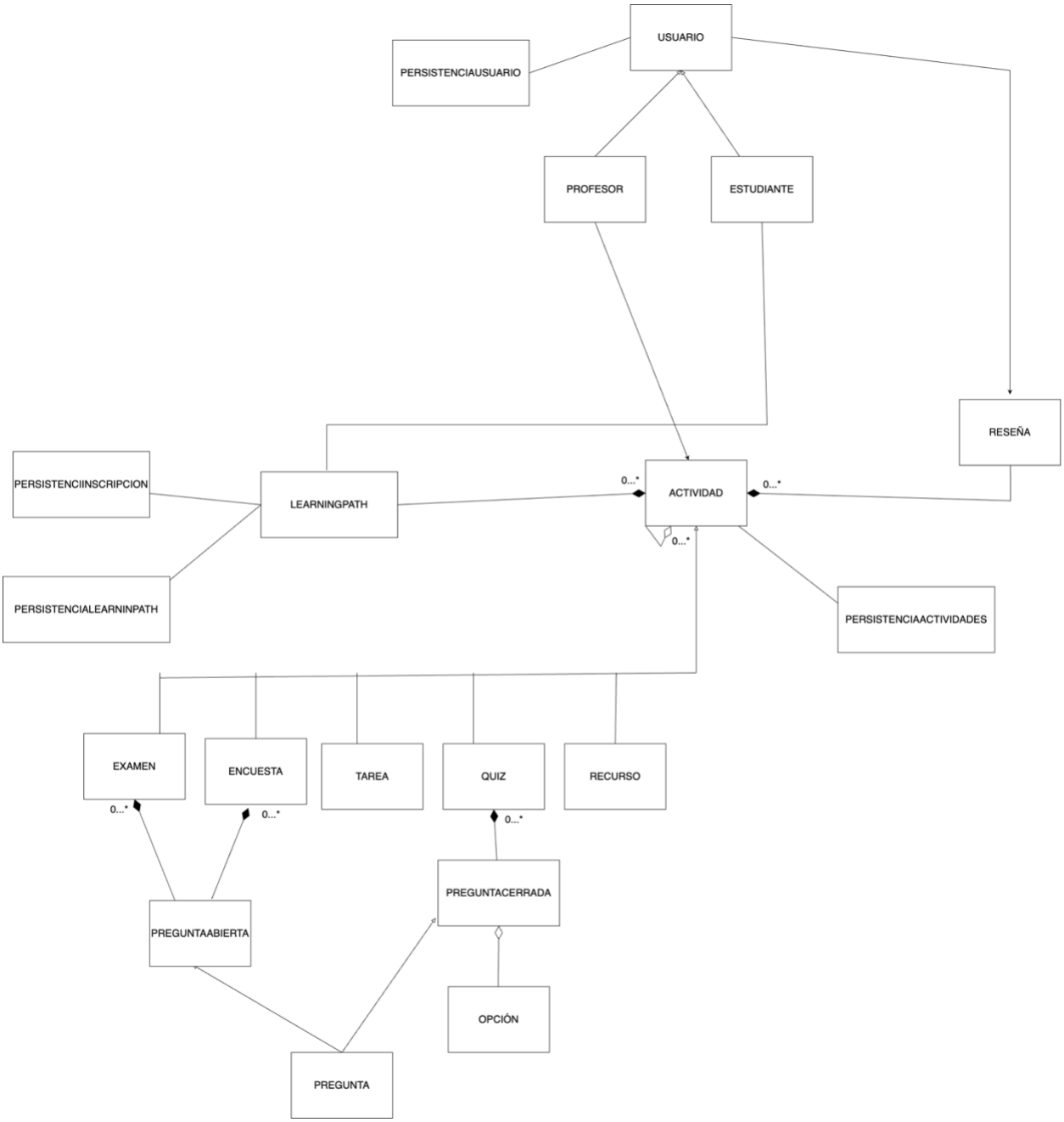
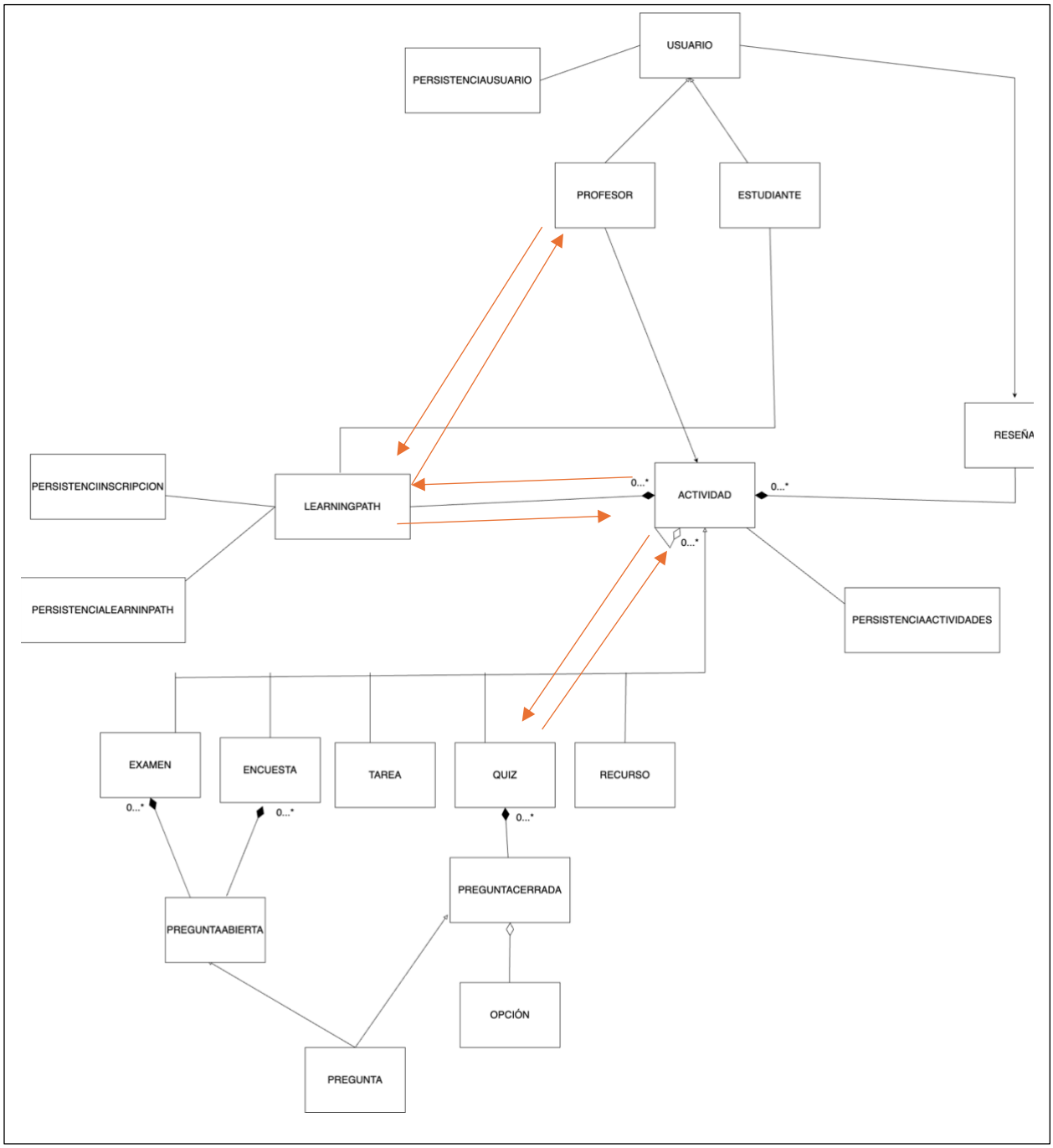

Usuario
- nombre: String
- correo: String
- password: String
+ Usuario(nombre: String, correo: String, password: String)
+ menu(): abstract void
+ getNombre(): String
+ setNombre(nombre: String): void
+ getCorreo(): String
+ setCorreo(correo: String): void
+ getPassword(): String
+ setPassword(password: String): void
PersistenciaUsuario
+ guardarUsuarioEnCSV(nombre: String, correo: String, password: String, tipoUsuario: String): void
+ cargarUsuariosDeCSV(): Map<String, Usuario>
Profesor
- learningpaths: ArrayList<LearningPath>
+ Profesor(nombre: String, correo: String, password: String)
+ menu(): void
+ mostrarLearningPaths(): void
+ crearQuiz(descripcion: String, objetivo: String, nivelDificultad: String, duracion: int, fechaLim: Date, obligatoria: boolean, calificacionMin: double): Actividad
+ crearRecurso(descripcion: String, objetivo: String, nivelDificultad: String, duracion: int, fechaLim: Date, obligatoria: boolean, tipoRecurso: String): Actividad
+ CrearExamen(): Actividad
+ addLearningPath(learningpath: LearningPath): void
+ getLearningpaths(): ArrayList<LearningPath>
+ setLearningpaths(learningpaths: ArrayList<LearningPath>): void
Estudiante
- learningPaths: List<LearningPath>
- actividadActual: Actividad
+ Estudiante(nombre: String, correo: String, password: String)
+ menu(): void
+ addLearningPath(learningPath: LearningPath): void
+ mostrarLearningPathsInscritos(): void
+ mostrarLearningPathsDisponibles(mapaLearnigPaths: Map<String, List<Actividad>>): void
+ iniciarActividad(actividad: Actividad): void
+ completarActividad(): void
+ añadirResenaActividad(actividad: Actividad, comentario: String, rating: int): void
Reseña
- cuerpo: String
+ Reseña(cuerpo: String)
+ getCuerpo(): String
+ setCuerpo(cuerpo: String): void
PersistenciaInscripciones
- RUTA_CSV: String
+ guardarInscripcion(correoEstudiante: String, nombreLearningPath: String): void
+ cargarInscripciones(usuarios: Map<String, Usuario>, learningPaths: Map<String, LearningPath>): void
LearningPath
- titulo: String
- descripcion: String
- objetivos: String
- nivelDificultad: String
- duracion: int
- rating: Double
- actividades: List<Actividad>
- actividadesCompletadas: List<Actividad>
- fechaCreacion: Date
- fechaUltModificacion: Date
+ LearningPath(titulo: String, descripcion: String, objetivos: String, nivelDificultad: String)
+ getTitulo(): String
+ setTitulo(titulo: String): void
+ getDescripcion(): String
+ setDescripcion(descripcion: String): void
+ getObjetivos(): String
+ setObjetivos(objetivos: String): void
+ getNivelDificultad(): String
+ setNivelDificultad(nivelDificultad: String): void
+ getDuracion(): int
+ setDuracion(duracion: int): void
+ getRating(): Double
+ setRating(rating: Double): void
+ getActividades(): List<Actividad>
+ setActividades(actividades: List<Actividad>): void
+ getFechaCreacion(): Date
+ setFechaCreacion(fechaCreacion: Date): void
+ getFechaUltModificacion(): Date
+ setFechaUltModificacion(fechaUltModificacion: Date): void
+ addActividad(actividad: Actividad): void
+ eliminarActividad(actividad: Actividad): void
+ completarActividad(actividad: Actividad): void
+ calcularProgreso(): void
+ mostrarProgreso(): void
+ mostrarActividades(): void
+ menu(): void
PersistenciaLearningPath
+ guardarLearningPath(learningPath: LearningPath, correo: String): void
+ cargarLearningPaths(usuarios: Map<String, Usuario>): void
Actividad
- tipo: String
+ descripcion: String
- objetivo: String
- nivelDificultad: String
- duracion: double
- actividadesPrerrequisito: List<Actividad>
- actividadesOpcionales: List<Actividad>
- fechaLim: Date
- estado: String
- completado: boolean
- obligatoria: boolean
- resenas: List<Reseña>
- ratings: List<Double>
- ratingProm: double
+ Actividad(tipo: String, descripcion: String, objetivo: String, nivelDificultad: String, duracion: double, fechaLim: Date, obligatoria: boolean)
+ getDescripcion(): String
+ setDescripcion(descripcion: String): void
+ getObjetivo(): String
+ setObjetivo(objetivo: String): void
+ getNivelDificultad(): String
+ setNivelDificultad(nivelDificultad: String): void
+ getDuracion(): double
+ setDuracion(duracion: double): void
+ getActividadesPrerrequisito(): List<Actividad>
+ setActividadesPrerrequisito(actividadesPrerrequisito: List<Actividad>): void
+ getFechaLim(): Date
+ setFechaLim(fechaLim: Date): void
+ getEstado(): String
+ setEstado(estado: String): void
+ isCompletado(): boolean
+ setCompletado(completado: boolean): void
+ getResenas(): List<Reseña>
+ setResenas(resenas: List<Reseña>): void
+ isObligatoria(): boolean
+ setObligatoria(obligatoria: boolean): void
+ addResena(resena: Reseña): void
+ addRating(rating: double): void
+ calcularProm(): void
+ completar(): abstract void
+ iniciar(): abstract void
PersistenciaActividades
+ guardarRecursosCSV(actividad: Actividad): void
+ cargarActividades(usuarios: Map<String, List<Actividad>>): void
+ convertirFechaStringYecha: Date): String
+ cargarActividadesSecundarias(datos: String[]): List<Actividad>
+ guardarActividadesSecundarias(actividad: Actividad, writer: FileWriter): void
+ isInteger(str: String): boolean
Examen
- preguntas: List<PreguntaAbierta>
+ Examen(descripcion: String, objetivo: String, nivelDificultad: String, duracion: double, fechaLim: Date, obligatoria: boolean, preguntas: List<PreguntaAbierta>)
+ getPreguntas(): List<PreguntaAbierta>
+ completar(): void
+ iniciar(): void
Encuesta
- preguntas: List<PreguntaAbierta>
+ Encuesta(descripcion: String, objetivo: String, nivelDificultad: String, duracion: double, fechaLim: Date, obligatoria: boolean, preguntas: List<PreguntaAbierta>)
+ getPreguntas(): List<PreguntaAbierta>
+ setPreguntas(preguntas: List<PreguntaAbierta>): void
+ addPregunta(pregunta: PreguntaAbierta): void
+ completar(): void
+ iniciar(): void
Tarea
+ Tarea(descripcion: String, objetivo: String, nivelDificultad: String, duracionFinal: double, fechaLim: Date, obligatoria: boolean, actividadesPrerrequisito: List<Actividad>, actividadesOpcionales: List<Actividad>)
+ completar(): void
+ iniciar(): void
Quiz
- preguntas: List<PreguntaCerrada>
- calificacionMin: double
+ Quiz(descripcion: String, objetivo: String, nivelDificultad: String, duracion: int, fechaLim: Date, calificacionMin: double, obligatoria: boolean)
+ addPregunta(pregunta: PreguntaCerrada): void
+ getPreguntas(): List<PreguntaCerrada>
+ setPreguntas(preguntas: List<PreguntaCerrada>): void
+ getCalificacionMin(): double
+ setCalificacionMin(calificacionMin: double): void
+ completar(): void
+ iniciar(): void
Recurso
- tipoRecurso: String
+ Recurso(descripcion: String, objetivo: String, nivelDificultad: String, duracion: double, fechaLim: Date, obligatoria: boolean, tipoRecurso: String)
+ getTipoRecurso(): String
+ setTipoRecurso(tipoRecurso: String): void
+ completar(): void
+ iniciar(): void
PreguntaAbierta
+ PreguntaAbierta(cuerpo: String)
+ getCuerpo(): String
PreguntaCerrada
- opciones: List<Opcion>
- opcionCorrecta: Opcion
+ PreguntaCerrada(cuerpo: String, opciones: List<Opcion>, opcionCorrecta: Opcion)
+ addOpcion(opcion: Opcion): void
+ getOpcionCorrecta(): Opcion
+ setOpcionCorrecta(opcionCorrecta: Opcion): void
+ getOpciones(): List<Opcion>
+ setOpciones(opciones: List<Opcion>): void
+ getCuerpo(): String
Opcion
- explicacion: String
+ Opcion(explicacion: String)
+ getExplicacion(): String
+ setExplicacion(explicacion: String): void
Pregunta
- cuerpo: String
+ Pregunta(cuerpo: String)
+ getCuerpo(): String
+ setCuerpo(cuerpo: String): void
0...*
0...*
0...*
0...*
0...*

DIAGRAMA B.


USUARIO
PERSISTENCIAUSUARIO
PROFESOR
ESTUDIANTE
RESEÑA
PERSISTENCIINSCRIPCION
LEARNINGPATH
ACTIVIDAD
0...*
0...*
0...*
PERSISTENCIALEARNINPATH
PERSISTENCIAACTIVIDADES
EXAMEN
ENCUESTA
TAREA
QUIZ
RECURSO
0...*
0...*
0...*
PREGUNTACERRADA
PREGUNTAABIERTA
OPCIÓN
PREGUNTA

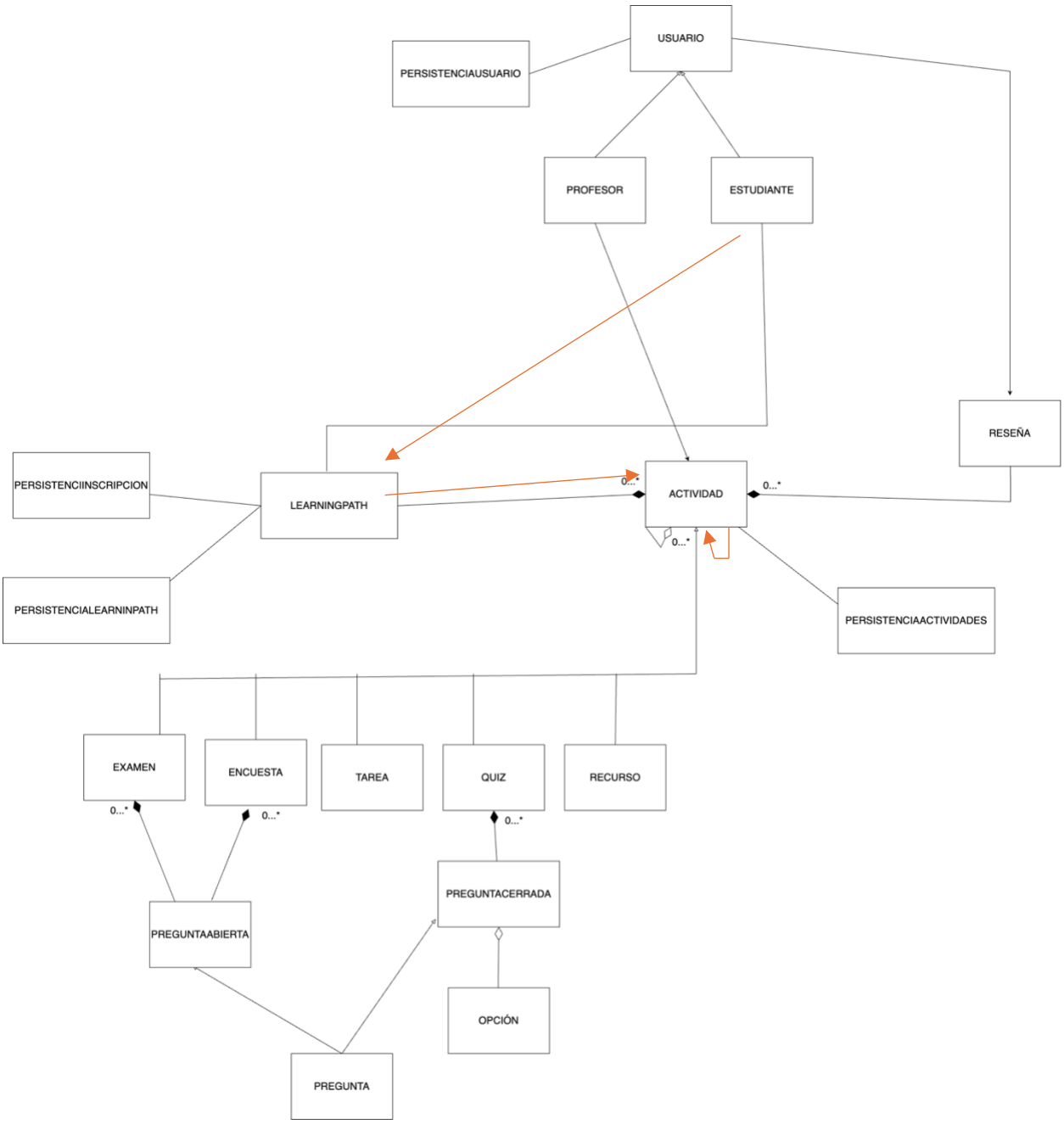
DIAGRAMAS C.



1. **El profesor solicita crear un Quiz.**
    - **Profesor envía un mensaje a LearningPath: crearQuiz(descripcion, objetivo, nivelDificultad, duracion, fechaLim, calificacionMin, obligatoria).**
2. **LearningPath crea el Quiz.**

- **LearningPath crea una nueva instancia de Quiz.**

3. **El Learning Path añade el Quiz a la lista de actividades.**
    - **LearningPath llama al método addActividad(quiz) para agregar el Quiz a su lista de actividades.**
4. **Confirmación al Profesor.**
    - **LearningPath envía un mensaje de confirmación al Profesor.**


USUARIO
PERSISTENCIAUSUARIO
PROFESOR
ESTUDIANTE
RESEÑA
PERSISTENCIINSCRIPCION
LEARNINGPATH
ACTIVIDAD
0...*
0...*
0...*
0...*
PERSISTENCIALEARNINPATH
PERSISTENCIAACTIVIDADES
EXAMEN
ENCUESTA
TAREA
QUIZ
RECURSO
0...*
0...*
0...*
PREGUNTACERRADA
PREGUNTAABIERTA
OPCIÓN
PREGUNTA

1. **El estudiante inicia la actividad.**
    - **Estudiante envía un mensaje a la Actividad: iniciarActividad().**
2. **La actividad cambia su estado a 'en progreso'.**
    - **La Actividad cambia su estado interno a "en progreso".**
3. **El estudiante completa la actividad.**
    - **Estudiante envía un mensaje a la Actividad: completarActividad().**
4. **La actividad cambia su estado a 'completada'.**
    - **La Actividad cambia su estado interno a "completada".**

Justificación de Decisiones de Diseño

Uso de Herencia

La clase Usuario es abstracta y permite que tanto Estudiante como Profesor extiendan de ella, lo que evita la duplicación de código. La herencia asegura que los usuarios comparten atributos comunes como nombre, correo, y password, mientras que las clases derivadas definen comportamientos específicos.

Composición

LearningPath contiene una lista de Actividad, lo que refleja una relación "parte-todo" fuerte. Este diseño garantiza que un Learning Path puede gestionar sus actividades de manera organizada y centralizada.

Separación de Responsabilidades (SRP)

Las clases de persistencia (PersistenciaUsuarios, PersistenciaLearningPaths) están separadas de las entidades principales. Esto sigue el principio de responsabilidad única, lo que facilita el mantenimiento del sistema y permite futuras mejoras sin impactar la lógica del dominio.

Extensibilidad

El uso de herencia en las clases Actividad y Pregunta permite que nuevas actividades o tipos de preguntas se agreguen al sistema sin modificar las clases existentes. Por ejemplo, agregar una nueva actividad como Simulación o una nueva pregunta como PreguntaMúltiple sería sencillo gracias a este diseño.