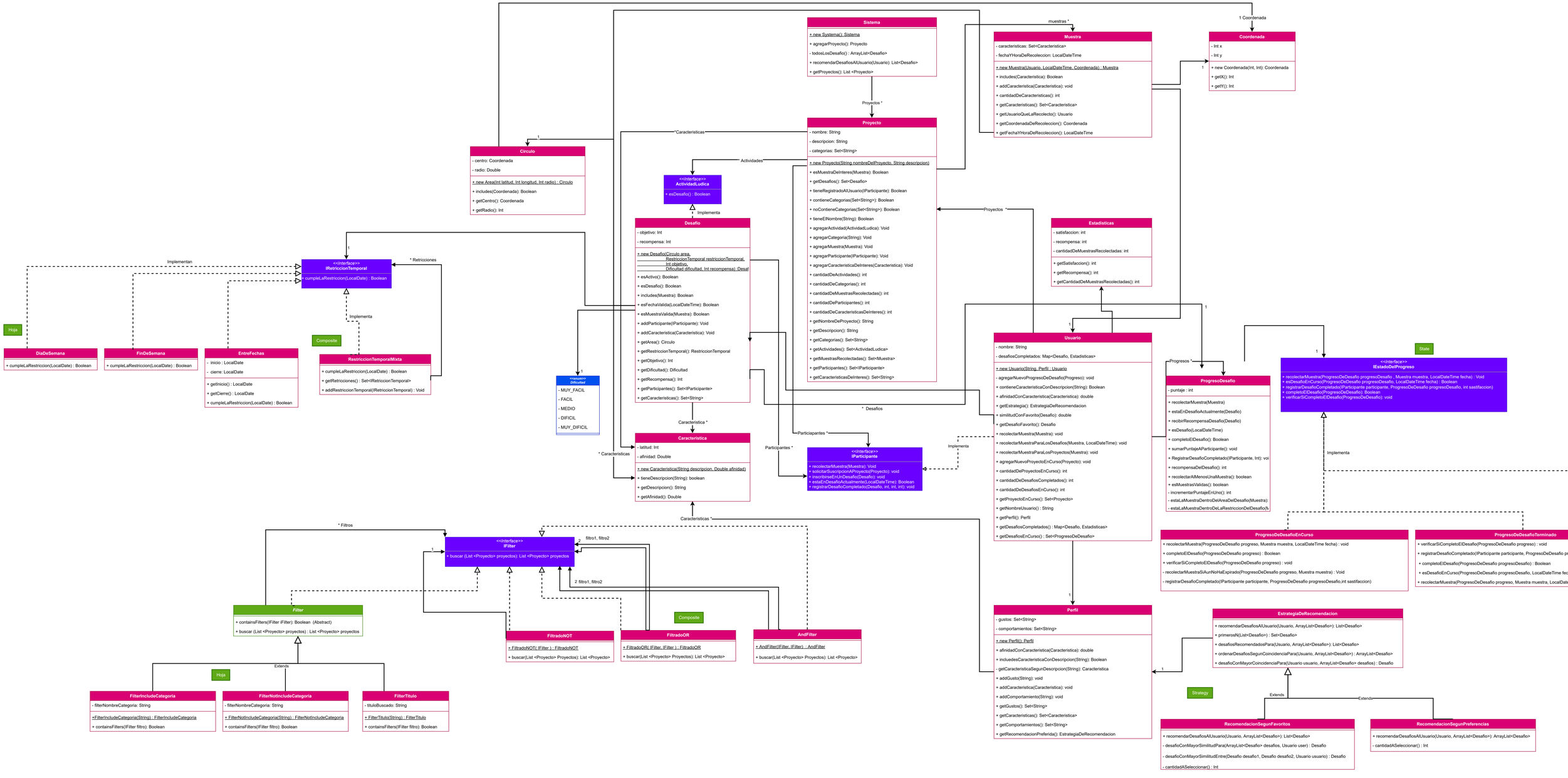
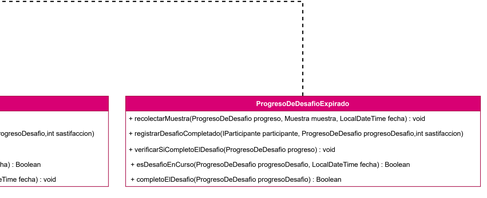

Sistema
+ new Systema(): Sistema
+ agregarProyecto(): Proyecto
- todosLosDesafio() : ArrayList<Desafio>
+ recomendarDesafiosAlUsuario(Usuario): List<Desafio>
+ getProyectos(): List <Proyecto>
Proyectos *
Proyecto
- nombre: String
- descripcion: String
- categorias: Set<String>
+ new Proyecto(String nombreDelProyecto, String descripcion)
+ esMuestraDeInteres(Muestra): Boolean
+ getDesafios(): Set<Desafio>
+ tieneRegistradoAlUsuario(IParticipante): Boolean
+ contieneCategorias(Set<String>): Boolean
+ noContieneCategorias(Set<String>): Boolean
+ tieneElNombre(String): Boolean
+ agregarActividad(ActividadLudica): Void
+ agregarCategoria(String): Void
+ agregarMuestra(Muestra): Void
+ agregarParticipante(IParticipante): Void
+ agregarCaracteristicaDeInteres(Caracteristica): Void
+ cantidadDeActividades(): int
+ cantidadDeCategorias(): int
+ cantidadDeMuestrasRecolectadas(): int
+ cantidadDeParticipantes(): int
+ cantidadDeCaracteristicasDeInteres(): int
+ getNombreDeProyecto(): String
+ getDescripcion(): String
+ getCategorias(): Set<String>
+ getActividades(): Set<ActividadLudica>
+ getMuestrasRecolectadas(): Set<Muestra>
+ getParticipantes(): Set<IParticipante>
+ getCaracteristicasDeInteres(): Set<String>
"Caracteristicas
" Actividades
Proyectos *
muestras *
Muestra
- caracteristicas: Set<Caracteristica>
- fechaYHoraDeRecoleccion: LocalDateTime
+ new Muestra(Usuario, LocalDateTime, Coordenada) : Muestra
+ includes(Caracteristica): Boolean
+ addCaracteristica(Caracteristica): void
+ cantidadDeCaracteristicas(): int
+ getCaracteristicas(): Set<Caracteristica>
+ getUsuarioQueLaRecolecto(): Usuario
+ getCoordenadaDeRecoleccion(): Coordenada
+ getFechaYHoraDeRecoleccion(): LocalDateTime
1 Coordenada
Coordenada
- Int x
- Int y
+ new Coordenada(Int, Int): Coordenada
+ getX(): Int
+ getY(): Int
Circulo
- centro: Coordenada
- radio: Double
+ new Area(Int latitud, Int longitud, Int radio) : Circulo
+ includes(Coordenada): Boolean
+ getCentro(): Coordenada
+ getRadio(): Int
<<Interface>>
ActividadLudica
+ esDesafio() : Boolean
Implementa
Desafio
- objetivo: Int
- recompensa: Int
+ new Desafio(Circulo area,
RestriccionTemporal restriccionTemporal,
Int objetivo,
Dificultad dificultad, Int recompensa) :Desaf
+ esActivo(): Boolean
+ esDesafio(): Boolean
+ includes(Muestra): Boolean
+ esFechaValida(LocalDateTime): Boolean
+ esMuestraValida(Muestra): Boolean
+ addParticipante(IParticipante): Void
+ addCaracteristica(Caracteristica): Void
+ getAreaa(): Circulo
+ getRestriccionTemporal(): RestriccionTemporal
+ getObjetivo(): Int
+ getDificultad(): Dificultad
+ getRecompensa(): Int
+ getParticipantes(): Set<IParticipante>
+ getCaracteristicas(): Set<String>
Caracteristica *
" Caracteristicas
Caracteristica
- latitud: Int
- afinidad: Double
+ new Caracteristica(String descripcion, Double afinidad)
+ tieneDescripcion(String): boolean
+ getDescripcion(): String
+ getAfinidad(): Double
Caracteristicas *
Participantes *
Particiapantes *
<<Interface>>
IParticipante
+ recolectarMuestra(Muestra): Void
+ solicitarSuscripcionAProyecto(Proyecto): void
+ inscribirseEnUnDesafio(Desafio): void
+ estaEnDesafioActualmente(LocalDateTime): Boolean
+ registrarDesafioCompletado(Desafio, int, int, int): void
Implementa
Estadisticas
- satisfaccion: int
- recompensa: int
- cantidadDeMuestrasRecolectadas: int
+ getSatisfaccion(): int
+ getRecompensa(): int
+ getCantidadDeMuestrasRecolectadas(): int
<<Interface>>
Dificultad
- MUY_FACIL
- FACIL
- MEDIO
- DIFICIL
- MUY_DIFICIL
" Desafios
Usuario
- nombre: String
- desafiosCompletados: Map<Desafio, Estadisticas>
+ new Usuario(String, Perfil) : Usuario
- agregarNuevoProgresoDeDesafio(Progreso): void
+ contieneCaracteristicaConDescripcion(String): Boolean
+ afinidadConCaracteristica(Caracteristica): double
+ getEstrategia(): EstrategiaDeRecomendacion
+ similitudConFavorito(Desafio): double
+ getDesafioFavorito(): Desafio
+ recolectarMuestra(Muestra): void
+ recolectarMuestraParaLosDesafios(Muestra, LocalDateTime): void
+ recolectarMuestraParaLosProyectos(Muestra): void
+ agregarNuevoProyectoEnCurso(Proyecto): void
+ cantidadDeProyectosEnCurso(): int
+ cantidadDeDesafiosCompletados(): int
+ cantidadDeDesafiosEnCurso(): int
+ getProyectoEnCurso(): Set<Proyecto>
+ getNombreUsuario() : String
+ getPerfil(): Perfil
+ getDesafiosCompletados() : Map<Desafio, Estadisticas>
+ getDesafiosEnCurso() : Set<ProgresoDeDesafio>
Progresos *
ProgresoDesafio
- puntaje : int
+ recolectarMuestra(Muestra)
+ estaEnDesafioActualmente(Desafio)
+ recibirRecompensaDesafio(Desafio)
+ esDesafio(LocalDateTime)
+ completeElDesafio(): Boolean
+ sumarPuntajeAParticipante(): void
+ RegistrarDesafioCompletado(IParticipante, Int): voi
+ recompensaDelDesafio(): int
+ recolectarAlMenosUnaMuestra(): boolean
+ esMuestrasValidas(): boolean
- incrementarPuntajeEnUno(): int
- estaLaMuestraDentroDelAreaDelDesafio(Muestra):
- estaLaMuestraDentroDeLaRestriccionDelDesafio(M
State
<<Interface>>
IEstadoDelProgreso
+ recolectarMuestra(ProgresoDeDesafio progresoDesafio , Muestra muestra, LocalDateTime fecha) : Void
+ esDesafioEnCursoProgresoDeDesafio progresoDesafio, LocalDateTime fecha) : Boolean
+ registrarDesafioCompletado(IParticipante participante, ProgresoDeDesafio progresoDesafio, int sastifaccion)
+ completeElDesafioProgresoDeDesafio() :Boolean
+ verificarSiCompleteElDesafioProgresoDeDesafio() :void
Implementa
ProgresoDeDesafioEnCurso
+ recolectarMuestra(ProgresoDeDesafio progreso, Muestra muestra, LocalDateTime fecha) : void
+ completeElDesafio(ProgresoDeDesafio progreso) : Boolean
+ verificarSiCompleteElDesafio(ProgresoDeDesafio progreso) : void
- recolectarMuestraSiAunNoHaExpirado(ProgresoDeDesafio progreso, Muestra muestra) : Void
- registrarDesafioCompletado(IParticipante participante, ProgresoDeDesafio progresoDesafio,int sastifaccion)
ProgresoDeDesafioTerminado
+ verificarSiCompleteElDesafio(ProgresoDeDesafio progreso) : void
+ registrarDesafioCompletado(IParticipante participante, ProgresoDeDesafio pr
+ completeElDesafio(ProgresoDeDesafio progresoDesafio) : Boolean
+ esDesafioEnCurso(ProgresoDeDesafio progresoDesafio, LocalDateTime fec
+ recolectarMuestra(ProgresoDeDesafio progreso, Muestra muestra, LocalDa
Perfil
- gustos: Set<String>
- comportamientos: Set<String>
+ new Perfil() : Perfil
+ afinidadConCaracteristica(Caracteristica): double
+ incluedesCaracteristicaConDescripcion(String): Boolean
- getCaracteristicaSegunDescripcion(String): Caracteristica
+ addGusto(String): void
+ addCaracteristica(Caracteristica): void
+ addComportamiento(String): void
+ getGustos(): Set<String>
+ getCaracteristicas(): Set<Caracteristica>
+ getComportamientos(): Set<String>
+ getRecomendacionPreferida(): EstrategiaDeRecomendacion
EstrategiaDeRecomendacion
+ recomendarDesafiosAlUsuario(Usuario, ArrayList<Desafio>): List<Desafio>
+ primerosN(List<Desafio>) : Set<Desafio>
+ desafiosRecomendadosPara(Usuario, ArrayList<Desafio>): List<Desafio>
+ ordenarDesafiosSegunCoincidenciaPara(Usuario, ArrayList<Desafio>) : ArrayList<Desafio>
+ desafioConMayorCoincidenciaPara(Usuario usuario, ArrayList<Desafio> desafios) : Desafio
Strategy
Extends
Extends
RecomendacionSegunFavoritos
+ recomendarDesafiosAlUsuario(Usuario, ArrayList<Desafio>): List<Desafio>
- desafioConMayorSimilitudPara(ArrayList<Desafio> desafios, Usuario user) : Desafio
- desafioConMayorSimilitudEntre(Desafio desafio1, Desafio desafio2, Usuario usuario) : Desafio
- cantidadASeleccionar() : Int
RecomendacionSegunPreferencias
+ recomendarDesafiosAlUsuario(Usuario, ArrayList<Desafio>): ArrayList<Desafio>
- cantidadASeleccionar() : Int
Hoja
DiaDeSemana
+ cumpleLaRestriccion(LocalDate) : Boolean
FinDeSemana
+ cumpleLaRestriccion(LocalDate) : Boolean
EntreFechas
- inicio : LocalDate
- cierre: LocalDate
+ getInicio() : LocalDate
+ getCierre() : LocalDate
+ cumpleLaRestriccion(LocalDate) : Boolean
Implementan
<<Interface>>
IRetriccionTemporal
cumpleLaRestriccion(LocalDate) : Boolean
* Retricciones
Implementa
Composite
RestriccionTemporalMixta
+ cumpleLaRestriccion(LocalDate) : Boolean
+ getRetricciones() : Set<IRetriccionTemporal>
+ addRestriccionTemporal(IRetriccionTemporal) : Void
* Filtros
<<Interface>>
IFilter
+ buscar (List <Proyecto> proyectos): List <Proyecto> proyectos
filtro1, filtro2
2 filtro1, filtro2
Filter
+ containsFilters(IFilter iFilter): Boolean (Abstract)
+ buscar (List <Proyecto> proyectos) : List <Proyecto> proyectos
Extends
Hoja
FilterIncludeCategoria
- filterNombreCategoria: String
+FilterIncludeCategoria(String) : FilterIncludeCategoria
+ containsFilters(IFilter filtro): Boolean
FilterNotIncludeCategoria
- filterNombreCategoria: String
+ FilterNotIncludeCategoria(String) : FilterNotIncludeCategoria
+ containsFilters(IFilter filtro): Boolean
FilterTitulo
- tituloBuscado: String
+ FilterTitulo(String) : FilterTitulo
+ containsFilters(IFilter filtro): Boolean
FiltradoNOT
+ FiltradoNOT(IFilter) : FiltradoNOT
+ buscar(List <Proyecto> Proyectos): List <Proyecto>
Composite
FiltradoOR
+ FiltradoOR(IFilter, IFilter) : FiltradoOR
+ buscar(List <Proyecto> Proyectos): List <Proyecto>
AndFilter
+ AndFilter(IFilter, IFilter) : AndFilter
+ buscar(List <Proyecto> Proyectos): List <Proyecto>


ogresoDesafio,int sastifaccion)
ha) : Boolean
Time fecha) : void

ProgresoDeDesafioExpirado
+ recolectarMuestra(ProgresoDeDesafio progreso, Muestra muestra, LocalDateTime fecha) : void
+ registrarDesafioCompletado(IParticipante participante, ProgresoDeDesafio progresoDesafio,int sastifaccion)
+ verificarSiCompletoElDesafio(ProgresoDeDesafio progreso) : void
+ esDesafioEnCurso(ProgresoDeDesafio progresoDesafio, LocalDateTime fecha) : Boolean
+ completoElDesafio(ProgresoDeDesafio progresoDesafio) : Boolean