

DIAGRAMAS DE REFERENCIA Y DIAGRAMAS GENERADOS

ID	Diagramas de Referencia (Humano)		Diagramas Generados (Herramienta)	
	Código en PlantUML	Diagrama en PlantUML	Código en PlantUML	Diagrama en PlantUML
HU-CL01	<pre>@startuml class Member { memberId : Int name : String email : String } class Profile { profileId : Int info : String tags : List<String> } Member "1" -- "1" Profile @enduml</pre>	<pre> classDiagram class Member { memberId : Int name : String email : String } class Profile { profileId : Int info : String tags : List<String> } Member "1" -- "1" Profile </pre> <p>The diagram shows two classes: Member and Profile. The Member class has attributes memberId (Int), name (String), and email (String). The Profile class has attributes profileId (Int), info (String), and tags (List<String>). A 1-to-1 relationship is shown between Member and Profile.</p>	<pre> @startuml class Member { + updateProfile(): void } class Profile { } class ProfileField { - name: String - value: String } Member "1" *-- "1" Profile Profile "1" *-- "*" ProfileField @enduml </pre>	<pre> classDiagram class Member { + updateProfile(): void } class Profile { } class ProfileField { - name: String - value: String } Member "1" *-- "1" Profile Profile "1" *-- "*" ProfileField </pre> <p>The generated diagram includes the Member class with an updateProfile() method. It also includes the Profile and ProfileField classes. The Member class has a 1-to-1 relationship with Profile, and Profile has a 1-to-many relationship with ProfileField. The ProfileField class contains attributes name and value.</p>

HU-CL02	<pre>@startuml class Practitioner { practitionerId : Int name : String } class Profile { profileId : Int experience : String } Practitioner "1" -- "1" Profile @enduml</pre>	<pre> classDiagram class Practitioner { practitionerId : Int name : String } class Profile { profileId : Int experience : String } Practitioner "1" .. "1" Profile </pre>	<pre>@startuml class Member { } class Practitioner { } class Profile { + addExperience(details: String): void } class Experience { - details: String } Practitioner --> Member Member "1" *-- "1" Profile Profile "1" *-- "*" Experience @enduml</pre>	<pre> classDiagram class Practitioner class Member class Profile { <<addExperience(details: String): void>> } class Experience { <<details: String>> } Practitioner --> Member Member "1" .. "1" Profile Profile "1" .. "*" Experience </pre>
---------	--	---	---	--

HU-CL03	<pre> @startuml class Trainer{ trainerId : Int name : String } class Profile { profileId : Int additional_info : String } Practitioner "1" -- "1" Profile @enduml </pre>	<pre> classDiagram class Trainer { trainerId : Int name : String } class Profile { profileId : Int additional_info : String } Trainer "1..1" --> "1..1" Profile </pre>	<pre> @startuml class Member { } class Trainer { + addProfileDetails(): void } class Profile { - details: String } class Page { } Trainer --> Member Member "1" -- "1" Profile Profile ..> Page @enduml </pre>	<pre> classDiagram class Trainer class Member class Profile { <<details: String>> } class Page Trainer "1..1" --> "1..1" Member Member "1..1" --> "1..1" Profile Profile ..> "1..1" Page </pre>
---------	--	---	--	---

HU-CL04	<pre>@startuml class Member { memberId : Int name : String email : String } class Profile { profileId : Int summary : String isPrivate : Boolean } class ProfileDirectory Member "1" -- "1" Profile : owns > ProfileDirectory ..> Profile : lists > @enduml</pre>	<pre> classDiagram class Member { memberId : Int name : String email : String } class Profile { profileId : Int summary : String isPrivate : Boolean } class ProfileDirectory Member "1" -- "1" Profile : owns ProfileDirectory ..> Profile : lists @enduml </pre> <p>The diagram shows three classes: Member, Profile, and ProfileDirectory. Member has attributes memberId, name, and email. Profile has attributes profileId, summary, and isPrivate. ProfileDirectory is associated with Profile via a dashed line labeled 'lists'. Member is associated with Profile via a solid line labeled 'owns'.</p>	<pre>@startuml class Member { + viewProfile() + connectWith() } class Profile {} class Connection {} Member "1" -- "1" Profile Member ..> Profile (Member, Member) .. Connection @enduml</pre>	<pre> classDiagram class Member { + viewProfile() + connectWith() } class Profile class Connection Member "1" -- "1" Profile Member ..> Profile (Member, Member) .. Connection @enduml </pre> <p>The diagram shows Member, Profile, and Connection classes. Member has methods viewProfile() and connectWith(). Profile is associated with Member via a solid line labeled 'owns'. Member is associated with Profile via a dashed line labeled 'lists'. Member is associated with Connection via a dashed line labeled 'viewProfile()' and 'connectWith()'.</p>
HU-CL05	<pre>@startuml class Member { memberId : Int name : String email : String } class Profile { profileId : Int name : String summary : String isPrivate : Boolean } class SearchCriteria { name : String tag : String location : String } class ProfileSearchService Member "1" -- "1" Profile : owns Profile ..> SearchCriteria : uses Profile ..> ProfileSearchService : returns matches @enduml</pre>	<pre> classDiagram class Member { memberId : Int name : String email : String } class Profile { profileId : Int name : String summary : String isPrivate : Boolean } class SearchCriteria { name : String tag : String location : String } class ProfileSearchService Member "1" -- "1" Profile : owns Profile ..> SearchCriteria : uses Profile ..> ProfileSearchService : returns matches @enduml </pre> <p>The diagram shows Member, Profile, SearchCriteria, and ProfileSearchService classes. Member has attributes memberId, name, and email. Profile has attributes profileId, name, summary, and isPrivate. SearchCriteria has attributes name, tag, and location. Profile is associated with Member via a solid line labeled 'owns'. Profile is associated with SearchCriteria via a dashed line labeled 'uses'. Profile is associated with ProfileSearchService via a dashed line labeled 'returns matches'.</p>	<pre>@startuml class Member { + searchProfiles(filter: SearchFilter): Profile[] } class Profile {} class ProfileField { - fieldName: String - fieldValue: String } class SearchFilter {} class Connection {} Member "1" -- "1" Profile Profile "1" *-- "*" ProfileField @enduml</pre>	<pre> classDiagram class Member { + searchProfiles(filter: SearchFilter): Profile[] } class Profile class ProfileField { - fieldName: String - fieldValue: String } class SearchFilter class Connection Member "1" -- "1" Profile Profile "1" *-- "*" ProfileField @enduml </pre> <p>The diagram shows Member, Profile, ProfileField, SearchFilter, and Connection classes. Member has method searchProfiles(filter: SearchFilter): Profile[]. Profile has attribute ProfileField. ProfileField has attributes fieldName and fieldValue. SearchFilter is associated with Profile via a dashed line labeled 'searchProfiles(filter: SearchFilter): Profile[]'. Profile is associated with ProfileField via a dashed line labeled '1' and '2'.</p>

	<pre> Member "1" -- "1" Profile : owns > ProfileSearchService ..> SearchCriteria : uses > ProfileSearchService ..> Profile : returns matches @enduml </pre>		<pre> Member ..> SearchFilter Member "2" -- "*" Connection @enduml </pre>	
HU-CL06	<pre> @startuml class Member { memberId : Int name : String email : String } class Profile { profileId : Int info: String isPrivate : Boolean } Member "1" -- "1" Profile @enduml </pre>	<pre> classDiagram class Member { memberId : Int name : String email : String } class Profile { profileId : Int info: String isPrivate : Boolean } Member "1" -- "1" Profile </pre>	<pre> @startuml class Member { } class Profile { + markAsPrivate(): void } class PrivacySetting { - visibility: String } Member "1" -- "1" Profile Profile "1" -- "1" PrivacySetting @enduml </pre>	<pre> classDiagram class Member class Profile { + markAsPrivate(): void } class PrivacySetting { - visibility: String } Profile "1" -- "1" PrivacySetting Profile "1" -- "1" Member </pre>

HU-CL07	<pre>@startuml class Member { memberId : Int name : String email : String } class Profile { profileId : Int info : String emailPrivate : Boolean } Member "1" -- "1" Profile @enduml</pre>	<pre> classDiagram class Member { memberId : Int name : String email : String } class Profile { profileId : Int info : String emailPrivate : Boolean } Member "1" -- "1" Profile </pre> <p>This diagram shows two classes: Member and Profile. Member has attributes memberId (Int), name (String), and email (String). Profile has attributes profileId (Int), info (String), and emailPrivate (Boolean). A one-to-one relationship exists between Member and Profile.</p>	<pre>@startuml class Member { } class Profile { + setEmailPrivacy(isPrivate: boolean) } class EmailAddress { - value: String } class PrivacySetting { - isPrivate: boolean } Member "1" -- "1" Profile Profile "1" -- "1" EmailAddress Profile "1" -- "1" PrivacySetting EmailAddress "1" -- "1" PrivacySetting @enduml</pre> <p>This diagram shows Member, Profile, EmailAddress, and PrivacySetting classes. Profile has a setEmailPrivacy method. EmailAddress has an attribute value. PrivacySetting has an attribute isPrivate. Relationships: Member to Profile (1..1), Profile to EmailAddress (1..1), Profile to PrivacySetting (1..1), and EmailAddress to PrivacySetting (1..1).</p>	<pre> classDiagram class Member class Profile { <<setEmailPrivacy(isPrivate: boolean)>> } class EmailAddress { <<value: String>> } class PrivacySetting { <<isPrivate: boolean>> } Member "1" -- "1" Profile Profile "1" -- "1" EmailAddress Profile "1" -- "1" PrivacySetting EmailAddress "1" -- "1" PrivacySetting </pre> <p>This diagram is identical to the previous one but includes the implementation details of the methods and attributes as annotations within the respective class boxes.</p>
HU-CL08	<pre>@startuml class Trainer { trainerId : Int name : String email : String } class Profile { profileId : Int info : String } class Course { courseId : Int title : String detailsURL : String }</pre>	<pre> classDiagram class Trainer { trainerId : Int name : String email : String } class Profile { profileId : Int info : String } class Course { courseId : Int title : String detailsURL : String } Trainer "1" -- "1" Profile Trainer "1" -- "0..1" Course </pre> <p>This diagram shows Trainer, Profile, and Course classes. Trainer has attributes trainerId (Int), name (String), and email (String). Profile has attribute profileId (Int). Course has attributes courseId (Int), title (String), and detailsURL (String). Relationships: Trainer to Profile (1..1), Trainer to Course (1..0..1).</p>	<pre>@startuml class Trainer { + listUpcomingClasses() } class Member { } class Profile { } class Course { + linkToDetailedPage: String } class CoursePage { } Trainer --> Member Member "1" -- "1" Profile Trainer "1" -- "*" Course Course "1" -- "1" CoursePage</pre> <p>This diagram shows Trainer, Member, Profile, Course, and CoursePage classes. Trainer has a listUpcomingClasses method. Course has a linkToDetailedPage attribute. Relationships: Trainer to Member (1..1), Member to Profile (1..1), Trainer to Course (1..*), and Course to CoursePage (1..1).</p>	<pre> classDiagram class Trainer { <<listUpcomingClasses()>> } class Member class Profile class Course { <<linkToDetailedPage: String>> } class CoursePage Trainer "1" -- "1" Member Member "1" -- "1" Profile Trainer "1" -- "*" Course Course "1" -- "1" CoursePage </pre> <p>This diagram is identical to the previous one but includes the implementation details of the methods and attributes as annotations within the respective class boxes.</p>

	<pre>Trainer "1" -- "1" Profile Trainer "1" -- "0..*" Course @enduml</pre>		<pre>@enduml</pre>	
HU-CL09	<pre>@startuml class CertificationCourse { courseId : Int title : String startDate : Date isOpen : Boolean } class CourseCatalog CourseCatalog ..> CertificationCourse : lists > @enduml</pre>	<pre> classDiagram class CourseCatalog class CertificationCourse { courseId : Int title : String startDate : Date isOpen : Boolean } CourseCatalog ..> CertificationCourse : lists </pre>	<pre> @startuml class Visitor { } class Course { -description: String -schedule: String } class CertificationCourse { } CertificationCourse --> Course Visitor ..> CertificationCourse @enduml </pre>	<pre> classDiagram class Visitor class Course { -description: String -schedule: String } class CertificationCourse CertificationCourse ..> Course </pre>
HU-CL10	<pre>@startuml class Event { eventId : Int name : String eventDate : Date location : String } class EventCatalog EventCatalog ..> Event : lists > @enduml</pre>	<pre> classDiagram class EventCatalog class Event { eventId : Int name : String eventDate : Date location : String } EventCatalog ..> Event : lists </pre>	<pre> @startuml class Visitor { } class Event { -name: String -date: DateTime -description: String } Visitor --> Event : views @enduml </pre>	<pre> classDiagram class Visitor class Event { -name: String -date: DateTime -description: String } Visitor --> Event : views </pre>