

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>	<pre> @startuml class Member { } class Profile { + updateDescription() } class ProfileField { - name: String - value: String } Member "1" *-- "1" Profile Profile "1" *-- "*" ProfileField @enduml </pre>	<pre> classDiagram class Member class Profile { updateDescription() } class ProfileField { name: String value: String } Member "1" -- "1" Profile Profile "1" *-- "*" ProfileField </pre>

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> <p>The diagram shows two classes: Practitioner and Profile. Practitioner has attributes practitionerId (Int) and name (String). Profile has attributes profileId (Int) and experience (String). A 1-to-1 relationship connects Practitioner and Profile.</p>	<pre>@startuml class Member { } class Practitioner { + addExperienceDetails(): void } class Profile { } class Experience { - details: String } Practitioner --> Member Practitioner "1" -- "1" Profile Profile "1" *-- "*" Experience @enduml</pre>	<pre> classDiagram class Member { } class Practitioner { + addExperienceDetails(): void } class Profile { } class Experience { - details: String } Practitioner --> Member Practitioner "1" -- "1" Profile Profile "1" *-- "*" Experience </pre> <p>The diagram shows four classes: Member, Practitioner, Profile, and Experience. Practitioner has an association role addExperienceDetails(). Member is associated with Practitioner. Profile is associated with Practitioner. Experience is associated with Profile via a multiplicity of 1..*. The Experience class has an attribute details (String).</p>
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" Profile </pre> <p>The diagram shows two classes: Trainer and Profile. Trainer has attributes trainerId (Int) and name (String). Profile has attributes profileId (Int) and additional_info (String). A 1-to-1 relationship connects Trainer and Profile.</p>	<pre>@startuml class Member { } class Trainer { + addProfileDetails(): void } class Profile { } class ProfilePage { } Trainer --> Member Member "1" -- "1" Profile ProfilePage "1" -- "1" Profile @enduml</pre>	<pre> classDiagram class Member { } class Trainer { + addProfileDetails(): void } class Profile { } class ProfilePage { } Trainer --> Member Member "1" -- "1" Profile ProfilePage "1" -- "1" Profile </pre> <p>The diagram shows four classes: Member, Trainer, Profile, and ProfilePage. Trainer has an association role addProfileDetails(). Member is associated with Trainer. Profile is associated with Member. ProfilePage is associated with Profile via a multiplicity of 1..1.</p>

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(member: Member): Profile + connectWith(member: Member): void } class Profile {} Member "1" -- "1" Profile Member "*" -- "*" Member : connections @enduml</pre>	<pre> classDiagram class Member { + viewProfile(member: Member): Profile + connectWith(member: Member): void } class Profile Member "*" -- "*" Member : connections </pre> <p>The diagram shows Member and Profile classes. Member has methods viewProfile and connectWith. Member is connected to Profile via a multiplicity asterisk (*) on both sides, labeled 'connections'.</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</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 </pre> <p>The diagram shows five classes: Member, Profile, SearchCriteria, ProfileSearchService, and a dashed-line class. Member has attributes memberId, name, and email. Profile has attributes profileId, name, summary, and isPrivate. SearchCriteria has attributes name, tag, and location. ProfileSearchService is associated with Profile via a dashed line labeled 'uses'.</p>	<pre>@startuml class Member { + searchProfiles(filters: SearchFilter): Profile[] + connectWith(member: Member): Connection } class Profile {} class ProfileField { - name: String - value: String } class SearchFilter {} class Connection {} Member "1" -- "1" Profile</pre>	<pre> classDiagram class Member { + searchProfiles(filters: SearchFilter): Profile[] + connectWith(member: Member): Connection } class Profile class SearchFilter class Connection Member "1" -- "1" Profile Member "1" -- "1" SearchFilter Profile "*" -- "*" Connection </pre> <p>The diagram shows Member, Profile, SearchFilter, and Connection classes. Member has methods searchProfiles and connectWith. Profile is connected to Connection via a multiplicity asterisk (*) on both sides. Member is connected to SearchFilter via a solid line labeled 'connectWith'.</p>

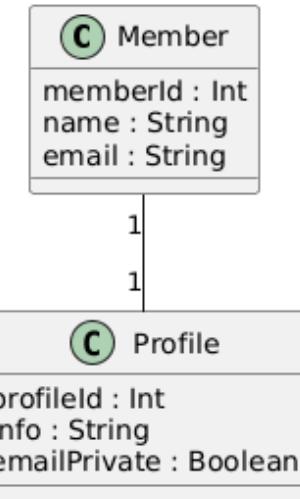
	<pre>> ProfileSearchService ..> SearchCriteria : uses > ProfileSearchService ..> Profile : returns matches @enduml</pre>		<p>Profile "1" *-- "*" ProfileField (Member, Member) .. Connection Member ..> SearchFilter @enduml</p>	
HU-CL06	<pre>@startuml class Member { memberId : Int name : String email : 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 { - name: String } Member "1" -- "1" Profile Profile "1" -- "1" PrivacySetting @enduml</pre>	<pre> classDiagram class Member class Profile { + markAsPrivate(): void } class PrivacySetting { - name: String } Member "1" .. "1" Profile Profile "1" .. "1" PrivacySetting </pre>

HU-CL07

```
@startuml
class Member {
    memberId : Int
    name : String
    email : String
}

class Profile {
    profileId : Int
    info : String
    emailPrivate : Boolean
}

Member "1" -- "1" Profile
@enduml
```



```
@startuml
class Member {
    + markEmailAsPrivate()
}

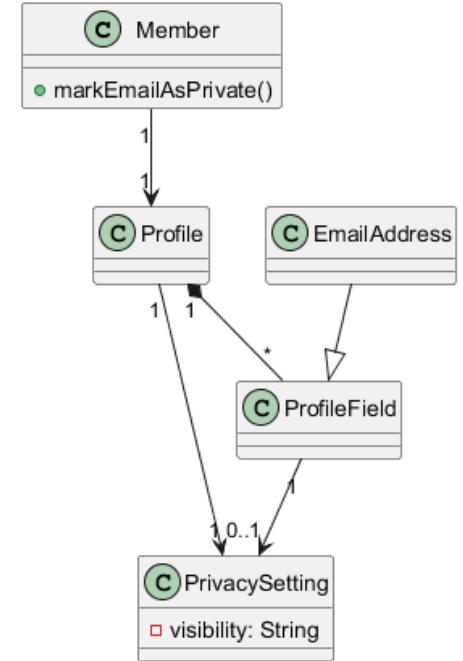
class Profile {
}

class PrivacySetting {
    - visibility: String
}

class ProfileField {
}

class EmailAddress {
}

Member "1" --> "1" Profile
Profile "1" *-- "*" ProfileField
EmailAddress --> ProfileField
Profile "1" --> "1" PrivacySetting
ProfileField "1" --> "0..1" PrivacySetting
@enduml
```



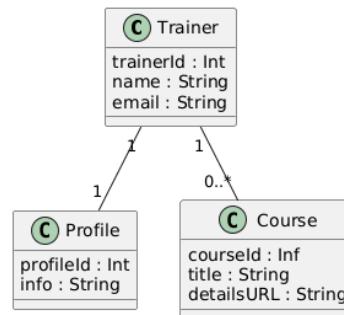
HU-CL08

```
@startuml
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..*" Course
@enduml
```



```
@startuml
class Trainer {
    + listUpcomingCourses()
}

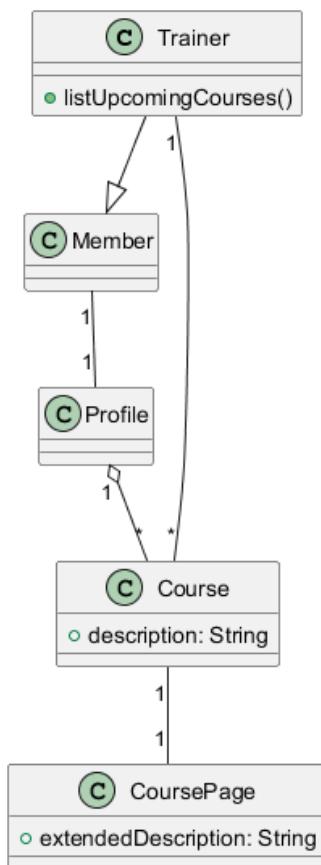
class Member {}

class Profile {}

class Course {
    + description: String
}

class CoursePage {
    + extendedDescription: String
}

Trainer --> Member
Member "1" -- "1" Profile
Trainer "1" -- "*" Course
Profile "1" o-- "*" Course
Course "1" -- "1" CoursePage
@enduml
```



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 "1..>" CertificationCourse : lists </pre> <p>The diagram shows a class hierarchy. At the top is a class labeled 'CourseCatalog' with a green circle containing a white 'C'. Below it is a class labeled 'CertificationCourse' with a green circle containing a white 'C'. A dashed arrow labeled 'lists' points from 'CourseCatalog' to 'CertificationCourse'.</p>	<pre>@startuml class Visitor { } class Course class CertificationCourse { } Visitor ..> CertificationCourse CertificationCourse --> Course @enduml</pre>	<pre> classDiagram class Visitor class CertificationCourse class Course Visitor ..> CertificationCourse CertificationCourse --> Course </pre> <p>This diagram illustrates the Visitor pattern. It features a 'Visitor' class at the top with a green circle containing a white 'C'. Below it is a 'CertificationCourse' class with a green circle containing a white 'C'. At the bottom is a 'Course' class with a green circle containing a white 'C'. A dashed arrow points from 'Visitor' to 'CertificationCourse', and another dashed arrow points from 'CertificationCourse' to 'Course'.</p>
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 "1..>" Event : lists </pre> <p>The diagram shows a class hierarchy. At the top is a class labeled 'EventCatalog' with a green circle containing a white 'C'. Below it is an 'Event' class with a green circle containing a white 'C'. A dashed arrow labeled 'lists' points from 'EventCatalog' to 'Event'.</p>	<pre>@startuml class Visitor { + viewUpcomingEvents() } class Event { - name: String - date: DateTime - description: String } Visitor ..> Event @enduml</pre>	<pre> classDiagram class Visitor class Event Visitor "+>" Event : viewUpcomingEvents() </pre> <p>This diagram illustrates the Visitor pattern. It features a 'Visitor' class at the top with a green circle containing a white 'C'. Below it is an 'Event' class with a green circle containing a white 'C'. A dashed arrow labeled '+>' points from 'Visitor' to 'Event', with the method name 'viewUpcomingEvents()' listed.</p>