

DIAGRAMAS DE REFERENCIA Y DIAGRAMAS GENERADOS

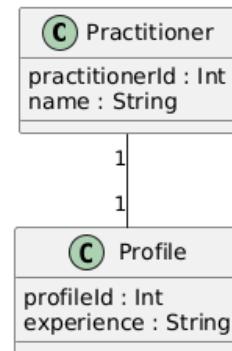
ID	Diagramas de Referencia (Humano)		Diagramas Generados (Herramienta)	
	Código en PlantUML	Diagrama en PlantUML	Código en PlantUML	Diagrama en PlantUML
HU-CL01	<pre>@startuml class Member { memberId : Int name : String email : String } class Profile { profileId : Int info : String tags : List<String> } 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 { - fieldName: String - fieldValue: String } class Page { } Member "1" *-- "1" Profile Profile "1" *-- "*" ProfileField Profile --> Page @enduml </pre>	<pre> classDiagram class Member class Profile { <<updateDescription()>> } class ProfileField { <<fieldName: String, fieldValue: String>> } class Page Member "1" -- "1" Profile Profile "1" *-- "*" ProfileField Profile --> Page </pre>

HU-CL02

```
@startuml
class Practitioner {
    practitionerId : Int
    name : String
}

class Profile {
    profileId : Int
    experience : String
}

Practitioner "1" -- "1" Profile
@enduml
```



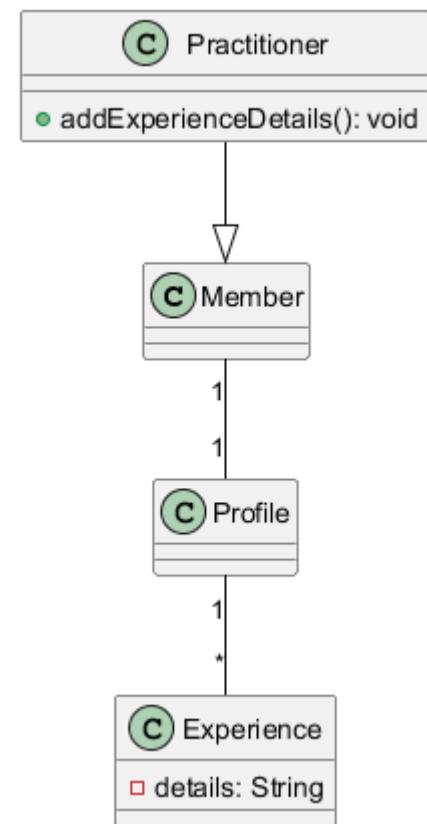
```
@startuml
class Member {
}

class Practitioner {
    + addExperienceDetails(): void
}

class Profile {
}

class Experience {
    - details: String
}

Practitioner --> Member
Member "1" -- "1" Profile
Profile "1" -- "*" Experience
@enduml
```

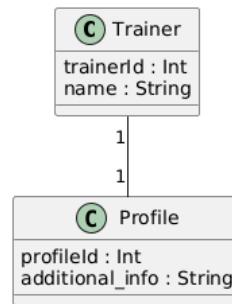


HU-CL03

```
@startuml
class Trainer{
    trainerId : Int
    name : String
}

class Profile {
    profileId : Int
    additional_info : String
}

Practitioner "1" -- "1" Profile
@enduml
```



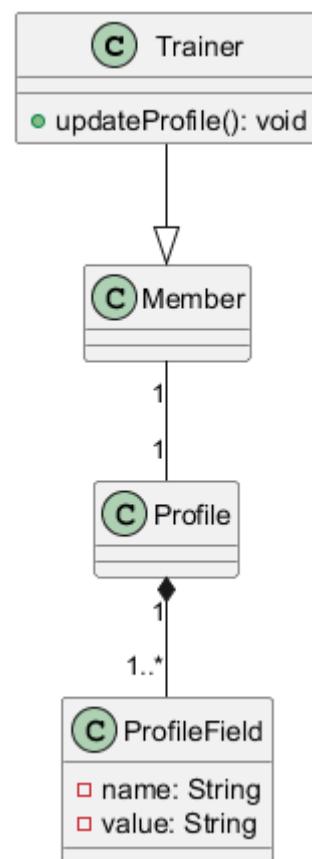
```
@startuml
class Trainer {
    + updateProfile(): void
}

class Member {}

class Profile {}

class ProfileField {
    - name: String
    - value: String
}

Trainer --> Member
Member "1" -- "1" Profile
Profile "1" *-- "1..*" ProfileField
@enduml
```



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 multiplicity of 1.. at Profile and .. at ProfileDirectory, labeled 'lists'. Member is associated with Profile via a multiplicity of 1.. at Profile and 1 at Member, labeled 'owns'.</p>	<pre>@startuml class Member { + viewProfile(profile: Profile): void + requestConnection(member: Member): void } class Profile {} class Connection {} Member "1" -- "1" Profile (Member, Member) .. Connection @enduml</pre>	<pre> classDiagram class Member { + viewProfile(profile: Profile): void + requestConnection(member: Member): void } class Profile {} class Connection {} Member "1" -- "1" Profile Member ..> Connection @enduml </pre> <p>The diagram shows Member, Profile, and Connection classes. Member has methods viewProfile and requestConnection. Profile and Connection are empty classes. Member is associated with Profile via a multiplicity of 1.. at Profile and 1 at Member, labeled 'owns'. Member is associated with Connection via a multiplicity of 1.. at Member and 1 at Connection, labeled 'lists'.</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 Member ..> SearchCriteria : uses ProfileSearchService ..> SearchCriteria : returns matches @enduml </pre> <p>The diagram shows five classes: Member, Profile, SearchCriteria, ProfileSearchService, and a fourth unnamed 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 SearchCriteria via a multiplicity of .. at ProfileSearchService and .. at SearchCriteria, labeled 'uses'. Member is associated with Profile via a multiplicity of 1.. at Profile and 1 at Member, labeled 'owns'. Member is associated with SearchCriteria via a multiplicity of .. at Member and .. at SearchCriteria, labeled 'uses'.</p>	<pre>@startuml class Member { + searchForProfiles(filter: SearchFilter): Profile[] + connectWith(member: Member): Connection } class Profile {} class ProfileField {} class SearchFilter {} class Connection {} Member "1" -- "1" Profile Profile "1" *-- "1..*" ProfileField Member ..> SearchFilter</pre>	<pre> classDiagram class Member { + searchForProfiles(filter: SearchFilter): Profile[] + connectWith(member: Member): Connection } class Profile {} class ProfileField {} class SearchFilter {} class Connection {} Member "1" -- "1" Profile Profile "1" *-- "1..*" ProfileField Member ..> SearchFilter @enduml </pre> <p>The diagram shows Member, Profile, ProfileField, SearchFilter, and Connection classes. Member has methods searchForProfiles and connectWith. Profile, ProfileField, and SearchFilter are empty classes. Member is associated with Profile via a multiplicity of 1.. at Profile and 1 at Member, labeled 'owns'. Profile is associated with ProfileField via a multiplicity of 1.. at Profile and 1.. at ProfileField, labeled 'lists'. Member is associated with SearchFilter via a multiplicity of .. at Member and .. at SearchFilter, labeled 'uses'.</p>

	<pre>> ProfileSearchService ..> SearchCriteria : uses > ProfileSearchService ..> Profile : returns matches @enduml</pre>		<pre>Member ..> Profile SearchFilter ..> ProfileField (Member, Member) .. 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 { + setProfilePrivacy(setting: PrivacySetting): void } class Profile { } class PrivacySetting { - visibility: String } Member "1" -- "1" Profile Profile "1" -- "1" PrivacySetting Member ..> PrivacySetting @enduml</pre>	<pre> classDiagram class Member { &gt; setProfilePrivacy(setting: PrivacySetting): void } class Profile class PrivacySetting { &lt; visibility: String } Member "1" ..> "1" PrivacySetting </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, name, and email. Profile has attributes profileId, info, and emailPrivate. A one-to-one relationship exists between Member and Profile.</p>	<pre>@startuml class Member { } class Profile { + setEmailPrivacy(isPrivate: boolean): void } 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>	<pre> classDiagram class Member class Profile { <<+ setEmailPrivacy(isPrivate: boolean): void>> } 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 includes the original classes and their relationships. It also adds three operations: setEmailPrivacy on Profile, value on EmailAddress, and isPrivate on PrivacySetting.</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 } Trainer "1" -- "1" Profile Trainer "1" -- "0..*" Course @enduml</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..*" Course </pre> <p>This diagram shows three classes: Trainer, Profile, and Course. Trainer has attributes trainerId, name, and email. Profile has attribute profileId and info. Course has attributes courseId, title, and detailsURL. Trainer has a one-to-one relationship with Profile and a zero-to-many relationship with Course.</p>	<pre>@startuml class Trainer { + listUpcomingClasses(): Course[] } class Profile { } class Course { - description: String - schedule: String } class CoursePage { - extendedDescription: String - link: String } Trainer "1" -- "1" Profile Trainer "1" -- "*" Course Course "1" -- "1" CoursePage @enduml</pre>	<pre> classDiagram class Trainer { <<+ listUpcomingClasses(): Course[]>> } class Profile class Course { <<- description: String>> <<- schedule: String>> } class CoursePage { <<- extendedDescription: String>> <<- link: String>> } Trainer "1" -- "1" Profile Trainer "1" -- "*" Course Course "1" -- "1" CoursePage </pre> <p>This diagram includes the original classes and their relationships. It also adds three operations: listUpcomingClasses on Trainer, description and schedule on Course, and extendedDescription and link on CoursePage.</p>

			@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" --> "3..>" CertificationCourse CourseCatalog < --> CertificationCourse </pre> <p>The diagram shows a CourseCatalog class with attributes <code>courseId</code>, <code>title</code>, <code>startDate</code>, and <code>isOpen</code>. It has a relationship named <code>lists</code> with multiplicity <code>3..></code> pointing to a CertificationCourse class. There is also a bidirectional relationship between CourseCatalog and CertificationCourse.</p>	<pre> @startuml class Visitor { + viewUpcomingCertificationCourses() } class Course { - name: String - description: String - schedule: String } class CertificationCourse { } CertificationCourse --> Course Visitor ..> CertificationCourse @enduml </pre> <p>The code defines a Visitor class with a method <code>viewUpcomingCertificationCourses()</code>. It also defines a Course class with attributes <code>name</code>, <code>description</code>, and <code>schedule</code>. A CertificationCourse class is defined. A directed association <code>--></code> exists from CertificationCourse to Course. A generalization relationship <code>..></code> exists from Visitor to CertificationCourse.</p>	<pre> classDiagram class Visitor class CertificationCourse class Course { # name: String # description: String # schedule: String } Visitor "1" --> "1" CertificationCourse CertificationCourse "1" --> "1" Course </pre> <p>This diagram represents the implementation of the Visitor pattern. It shows a Visitor class with a method <code>viewUpcomingCertificationCourses()</code>. It points to a CertificationCourse object, which then points to a Course object. The Course object has three private attributes: <code>name</code>, <code>description</code>, and <code>schedule</code>.</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" --> "3..>" Event EventCatalog < --> Event </pre> <p>The diagram shows an EventCatalog class with a relationship named <code>lists</code> with multiplicity <code>3..></code> pointing to an Event class. There is also a bidirectional relationship between EventCatalog and Event.</p>	<pre> @startuml class Visitor { + viewUpcomingEvents(): List<Event> } class Event { - name: String - date: Date - description: String } class Course { } class CertificationCourse { } Visitor ..> Event Course --> Event CertificationCourse --> Course </pre> <p>The code defines a Visitor class with a method <code>viewUpcomingEvents(): List<Event></code>. It also defines an Event class with attributes <code>name</code>, <code>date</code>, and <code>description</code>. Course and CertificationCourse classes are defined. A generalization relationship <code>..></code> exists from Visitor to Event. A directed association <code>--></code> exists from Course to Event. A directed association <code>--></code> exists from CertificationCourse to Event.</p>	<pre> classDiagram class Visitor class Event class Course class CertificationCourse Visitor "1" --> "1" Event Course "1" --> "1" Event CertificationCourse "1" --> "1" Event </pre> <p>This diagram represents the implementation of the Visitor pattern for events. It shows a Visitor class with a method <code>viewUpcomingEvents(): List<Event></code>. It points to an Event object, which is then pointed to by both Course and CertificationCourse objects.</p>

			@enduml	
--	--	--	---------	--