

## **DIAGRAMAS DE REFERENCIA Y DIAGRAMAS GENERADOS**

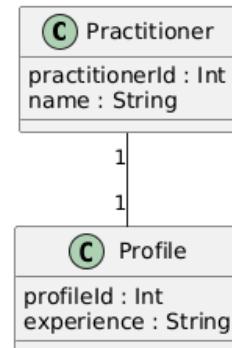
ID	Diagramas de Referencia		Diagramas Generados	
	Código en PlantUML	Diagrama en PlantUML	Código en PlantUML	Diagrama en PlantUML
HU-CL01	<pre>@startuml class Member {     memberId : Int     name : String     email : String }  class Profile {     profileId : Int     info : String     tags : List&lt;String&gt; }  Member "1" -- "1" Profile @enduml</pre>	<pre> classDiagram     class Member {         memberId : Int         name : String         email : String     }     class Profile {         profileId : Int         info : String         tags : List&lt;String&gt;     }     Member "1" -- "1" Profile </pre>	<pre> classMember {     memberId : Int     name : String     email : String } classProfile { } classPage { } classProfileField {     -name: String     -value: String } classVisitor {     +learnAboutMember() }  Member "1" *-- "1" Profile Profile --&gt; Page Profile "1" *-- "*" ProfileField Visitor ..&gt; Profile @enduml </pre>	

HU-CL02

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

class Profile {
    profileId : Int
    experience : String
}

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



```
@startuml
class Member {
}

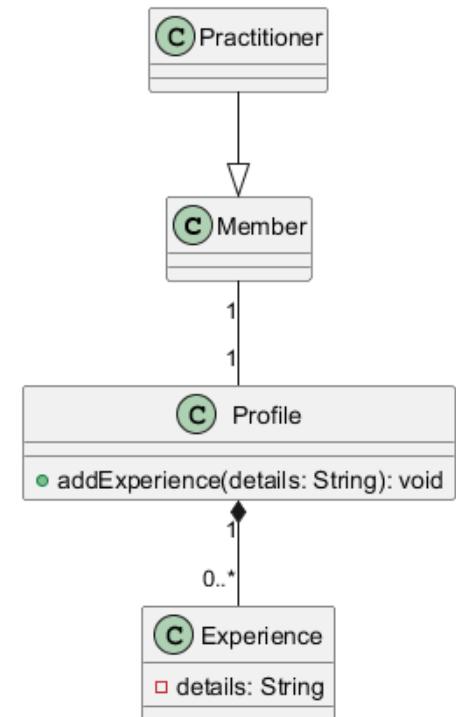
class Practitioner {
}

class Profile {
    + addExperience(details: String): void
}

class Experience {
    - details: String
}

Practitioner --> Member
Member "1" -- "1" Profile
Profile "1" *-- "0..*" Experience

@enduml
```



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>	<pre>@startuml class Member { }  class Trainer {     + addProfileDetails(): void }  class Profile { }  class ProfileField {     - title: String     - details: String }  Trainer --&gt; Member Member "1" -- "1" Profile Profile "1" *-- "*" ProfileField @enduml</pre>	<pre> classDiagram     class Trainer {         + addProfileDetails(): void     }     class Member     class Profile     class ProfileField {         □ title: String         □ details: String     }     Trainer "1" .. "1" Member     Member "1" .. "1" Profile     Profile "1" .. "*" ProfileField   </pre>
HU-CL04	<pre>@startuml class Member {     memberId : Int     name : String     email : String }  class Profile {     profileId : Int     summary : String     isPrivate : Boolean }  class ProfileDirectory</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     ProfileDirectory *-- "*" Profile   </pre>	<pre>@startuml class Member {     + requestConnection(member: Member): void }  class Profile { }  class PrivacySetting { }  Member "1" -- "1" Profile : has Member "*" -- "*" Member : has connection</pre>	<pre> classDiagram     class Member {         + requestConnection(member: Member): void     }     class Profile     class PrivacySetting     Member "1" .. "1" Profile     PrivacySetting *.. "*" Profile   </pre>

	<pre>Member "1" -- "1" Profile : owns &gt; ProfileDirectory ..&gt; Profile : lists &gt; @enduml</pre>		<p>with</p> <pre>Profile "1" -- "*" PrivacySetting : uses @enduml</pre>	
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 &gt; ProfileSearchService ..&gt; SearchCriteria : uses &gt; ProfileSearchService ..&gt; Profile : 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     ProfileSearchService ..&gt; SearchCriteria : uses &gt;     ProfileSearchService ..&gt; Profile : returns matches   </pre>	<pre>@startuml class Member {     + searchProfiles(criteria: ProfileField[]): Profile[] }  class Profile {  }  class ProfileField {     - fieldName: String     - fieldValue: String }  class Connection {  }  Member "1" -- "1" Profile Profile "1" *-- "1..*" ProfileField Member "2" -- "0..*" Connection @enduml</pre>	<pre> classDiagram     class Member {         + searchProfiles(criteria: ProfileField[]): Profile[]     }     class Profile {     }     class ProfileField {         - fieldName: String         - fieldValue: String     }     class Connection {     }      Member "1" -- "1" Profile     Profile "*" -- "1..*" ProfileField     Member "2" -- "0..*" Connection   </pre>

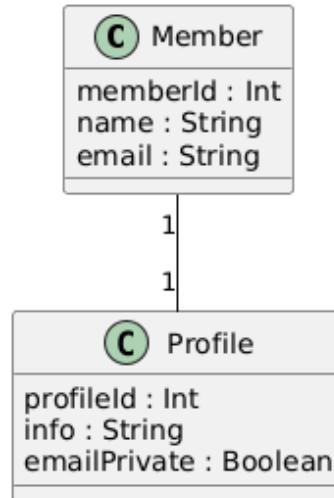
HU-CL06	<pre>@startuml class Member {     memberId : Int     name : String     email : String }  class Profile {     profileId : Int     info: String     isPrivate : Boolean } Member "1" -- "1" Profile @enduml</pre>	<pre> classDiagram     class Member {         memberId : Int         name : String         email : String     }     class Profile {         profileId : Int         info: String         isPrivate : Boolean     }     Member "1" -- "1" Profile </pre> <p>The 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 isPrivate (Boolean). A 1-to-1 relationship is shown between Member and Profile.</p>	<pre>@startuml class Member {     + setProfilePrivacy(setting: PrivacySetting): void }  class Profile { }  class PrivacySetting {     - visibilityLevel: String }  Member "1" -- "1" Profile Profile "1" -- "1" PrivacySetting Member ..&gt; PrivacySetting @enduml</pre>	<pre> classDiagram     class Member {         + setProfilePrivacy(setting: PrivacySetting): void     }     class Profile {     }     class PrivacySetting {         - visibilityLevel: String     }      Member "1" -- "1" Profile     Profile "1" -- "1" PrivacySetting     Member ..&gt; PrivacySetting </pre> <p>The diagram shows three classes: Member, Profile, and PrivacySetting. Member has a method setProfilePrivacy(setting: PrivacySetting). Profile has no methods. PrivacySetting has an attribute visibilityLevel. A 1-to-1 relationship exists between Member and Profile, and another 1-to-1 relationship exists between Profile and PrivacySetting. Member also has a generalization relationship to PrivacySetting.</p>
---------	---	---	---	---

HU-CL07

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

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

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



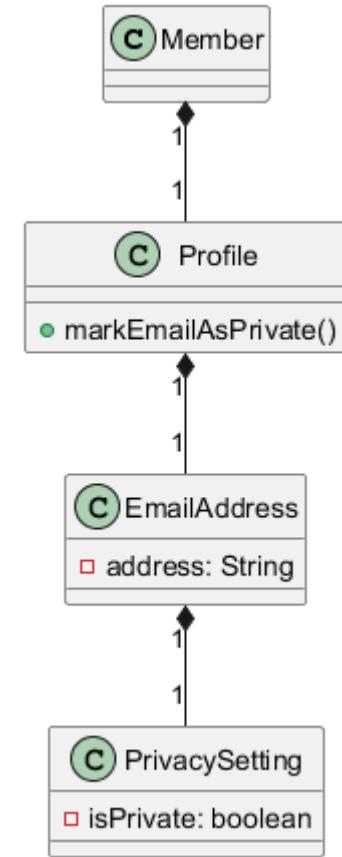
```
@startuml
class Member {
}
```

```
class Profile {
    + markEmailAsPrivate()
```

```
class EmailAddress {
    - address: String
}
```

```
class PrivacySetting {
    - isPrivate: boolean
}
```

```
Member "1" *-- "1" Profile
Profile "1" *-- "1" EmailAddress
EmailAddress "1" *-- "1" PrivacySetting
@enduml
```



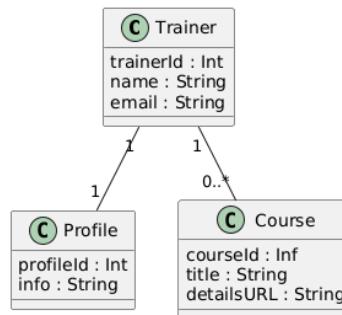
HU-CL08

```
@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 Member {
}

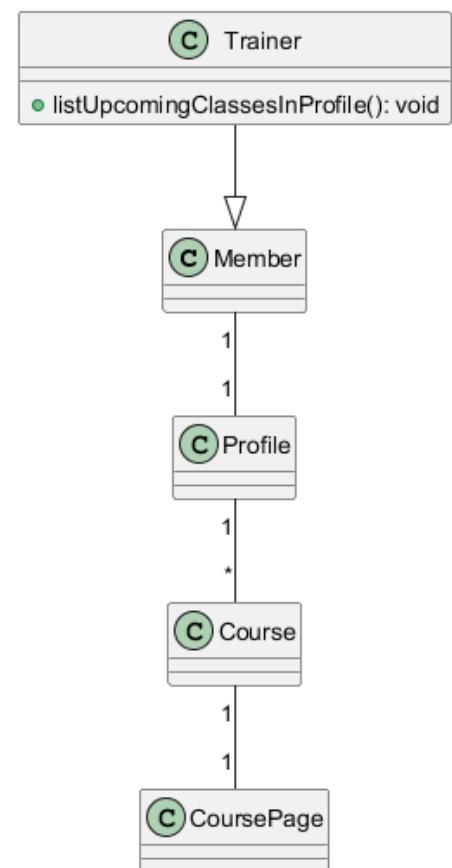
class Trainer {
    + listUpcomingClassesInProfile(): void
}

class Profile {
}

class Course {
}

class CoursePage {
}

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



HU-CL09	<pre>@startuml class CertificationCourse {     courseId : Int     title : String     startDate : Date     isOpen : Boolean }  class CourseCatalog  CourseCatalog ..&gt; CertificationCourse : lists &gt; @enduml</pre>	<pre> classDiagram     class CourseCatalog     class CertificationCourse {         courseId : Int         title : String         startDate : Date         isOpen : Boolean     }     CourseCatalog "1..&gt;" "lists" CertificationCourse </pre> <p>The diagram shows a class hierarchy. At the top is a class box labeled 'CourseCatalog'. Below it is another class box labeled 'CertificationCourse' with attributes: courseId (Int), title (String), startDate (Date), and isOpen (Boolean). A dashed arrow labeled 'lists' points from 'CourseCatalog' to 'CertificationCourse', indicating a one-to-many relationship.</p>	<pre>@startuml class Visitor { }  class Course {     -description: String     -schedule: String }  class CertificationCourse { }  CertificationCourse --&gt; Course Visitor ..&gt; "*" CertificationCourse @enduml</pre>	<pre> classDiagram     class Visitor     class Course {         #description: String         #schedule: String     }     class CertificationCourse      Visitor "*"--&gt; "*" CertificationCourse     CertificationCourse --&gt; Course </pre> <p>This diagram illustrates the Visitor design pattern. It features a 'Visitor' class at the top, followed by a 'Course' class with attributes description (String) and schedule (String). Below is a 'CertificationCourse' class. A multiplicity asterisk (*) is placed between 'Visitor' and 'CertificationCourse', and another (*) is placed between 'CertificationCourse' and 'Course', indicating that many visitors can visit many certification courses, and each course can be visited by many visitors.</p>
HU-CL10	<pre>@startuml class Event {     eventId : Int     name : String     eventDate : Date     location : String }  class EventCatalog  EventCatalog ..&gt; Event : lists &gt; @enduml</pre>	<pre> classDiagram     class EventCatalog     class Event {         eventId : Int         name : String         eventDate : Date         location : String     }     EventCatalog "1..&gt;" "lists" Event </pre> <p>The diagram shows a class hierarchy. At the top is a class box labeled 'EventCatalog'. Below it is another class box labeled 'Event' with attributes: eventId (Int), name (String), eventDate (Date), and location (String). A dashed arrow labeled 'lists' points from 'EventCatalog' to 'Event', indicating a one-to-many relationship.</p>	<pre>@startuml class Visitor {     +viewUpcomingEvents(): Event[] }  class Event {     -name: String     -date: Date     -description: String }  Visitor ..&gt; Event @enduml</pre>	<pre> classDiagram     class Visitor {         +viewUpcomingEvents(): Event[]     }     class Event {         #name: String         #date: Date         #description: String     }      Visitor --&gt; Event </pre> <p>This diagram illustrates the Visitor design pattern. It features a 'Visitor' class at the top with a method 'viewUpcomingEvents() returning Event[]'. Below is an 'Event' class with attributes name (String), date (Date), and description (String). A dashed arrow points from 'Visitor' to 'Event', indicating that the visitor can traverse the list of events.</p>