Datamodel

- Opleidingen
- Opleidingsresultaten
- Evaluaties
- Correspondentie API-datamodel en Pynter-datamodel

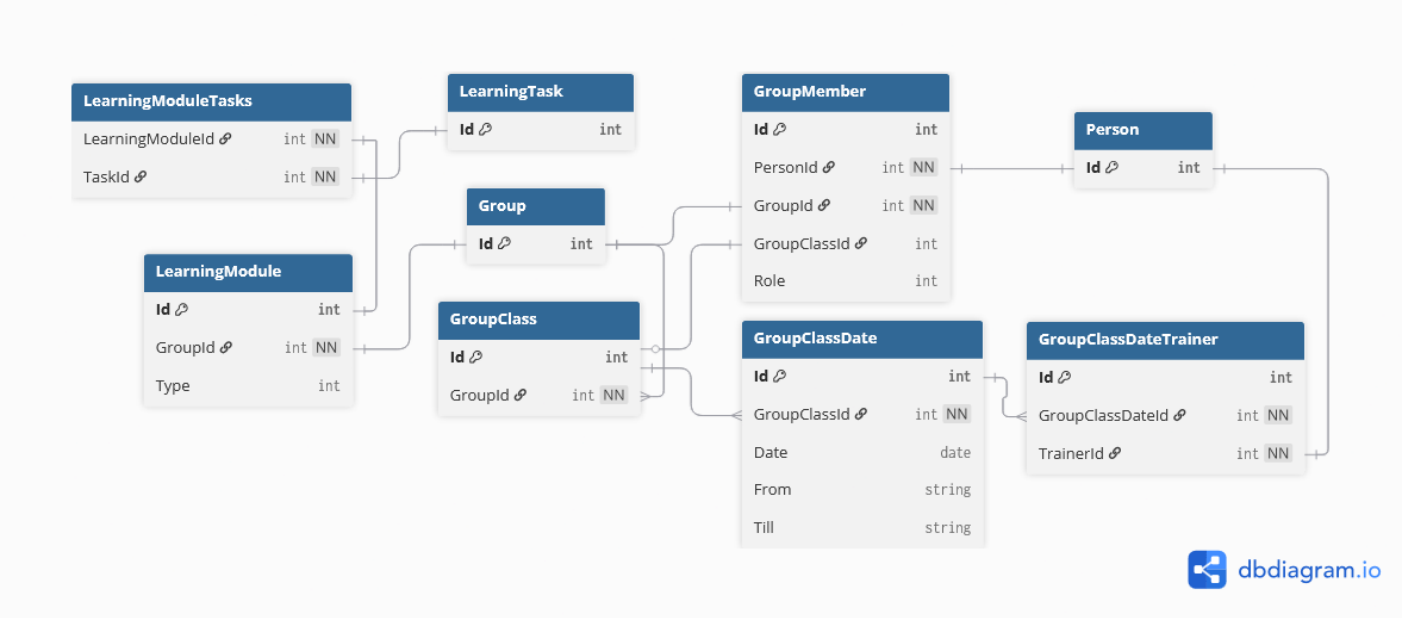
Opleidingen

Opleidingen in Pynter zijn modulair opgebouwd. Aan een opleiding (`LearningModule`) kunnen modules (`LearningTasks`) worden gekoppeld middels de koppeltabel `LearningModuleTasks`. Modules bevatten de verschillende soorten leeraanbod die op Pynter gevolgd kunnen worden, zoals LTI, SCORM en Pynter's eigen e-learning. Daarnaast kunnen modules ook gebruikt worden om andere gegevens bij te houden. Je kunt bijvoorbeeld een module maken waar een formulier moet worden ingevuld, of waarmee aanwezigheid kan worden geregistreerd.

Binnen Pynter is er onderscheid tussen opleidingen mét en zonder fysieke bijeenkomst. Wanneer een opleiding (deels) op locatie is, kunnen deze fysieke bijeenkomsten in Pynter worden geregistreerd. Pynter gebruikt hiervoor *lichtingen* (`GroupClasses`). Een lichting is een groep mensen die allemaal aanwezig is bij dezelfde trainingsmomenten. Per lichting kunnen een of meerdere trainingsdagen (`GroupClassDates`) worden toegevoegd, waarop kan worden aangegeven wanneer dit trainingsmoment plaatsvindt. Ook kan er per trainingsdag aangegeven worden wie die dag de opleiding verzorgt.

Voor het inschrijven voor een training gebruikt Pynter op de achtergrond groepen (`Groups`). Elke opleiding is gekoppeld aan een groep, en wanneer iemand zich inschrijft voor een opleiding wordt deze persoon ook automatisch lid van de groep (`GroupMember`). Wanneer iemand zich inschrijft voor een lichting, wordt naast de `GroupId` ook de `GroupClassId` vastgelegd.

Verder is het ook nog mogelijk om beheerders en trainers voor de gehele opleiding aan te wijzen. Dit gebeurt door mensen lid te maken van de groep met een specifieke rol (`GroupMember.Role`).



Broncode diagram

Dit is de broncode van het diagram hierboven. Dit kun je gebruiken om het diagram te bewerken op dbdiagram.io.

```
table Person {
  Id int [pk]
}

table LearningModule {
  Id int [pk]
  GroupId int [ref: - Group.Id, not null]
  Type int
}

table LearningModuleTasks {
  LearningModuleId int [ref: - LearningModule.Id, not null]
  TaskId int [ref: - LearningTask.Id, not null]
}

table LearningTask {
  Id int [pk]
}

table Group {
  Id int [pk]
}

table GroupClass {
  Id int [pk]
  GroupId int [ref: > Group.Id, not null]
}

table GroupClassDate {
  Id int [pk]
  GroupClassId int [ref: > GroupClass.Id, not null]
  Date date
  From string
  Till string
}
```

```
table GroupClassDateTrainer {
  Id int [pk]
  GroupClassDateId int [ref: > GroupClassDate.Id, not null]
  TrainerId int [ref: - Person.Id, not null]
}

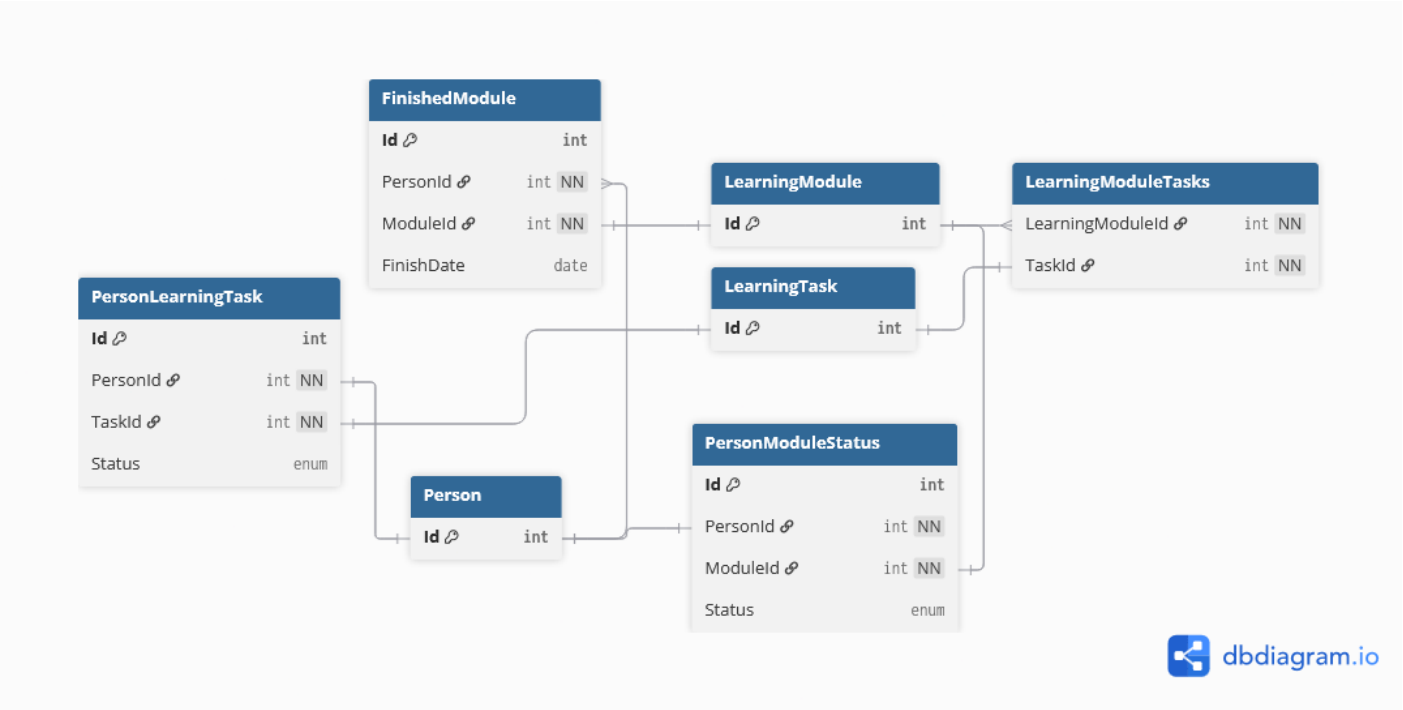
table GroupMember {
  Id int [pk]
  PersonId int [ref: - Person.Id, not null]
  GroupId int [ref: - Group.Id, not null]
  GroupClassId int [ref: - GroupClass.Id, null]
  Role int
}
```

Opleidingsresultaten

Wanneer een cursist een opleiding afrondt wordt dit in Pynter geregistreerd in de tabel `FinishedModule`. Dit noemen we een *resultaat*. Hierin wordt (o.a.) geregistreerd wanneer iemand welke opleiding heeft afgerond. Belangrijk detail hierbij is dat het mogelijk is om per opleiding meerdere resultaten te hebben, bijvoorbeeld wanneer een opleiding periodiek herhaald moet worden.

Om gemakkelijker inzicht te krijgen in de *huidige* status van een opleiding voor een persoon is er de tabel `PersonModuleStatus`. Waar een opleiding in de `FinishedModule` meerdere keren kan voorkomen, bevat de `PersonModuleStatus` altijd één rij per persoon per opleiding, als deze persoon ooit aan deze opleiding begonnen is.

Verder wordt er ook per persoon per module (`LearningTask`) bijgehouden wat de status daarvan is.



Broncode diagram

Dit is de broncode van het diagram hierboven. Dit kun je gebruiken om het diagram te bewerken op dbdiagram.io.

```
table Person {
  Id int [pk]
}
```

```
table LearningModule {
  Id int [pk]
}

table LearningModuleTasks {
  LearningModuleId int [ref: > LearningModule.Id, not null]
  TaskId int [ref: - LearningTask.Id, not null]
}

table LearningTask {
  Id int [pk]
}

table PersonLearningTask {
  Id int [pk]
  PersonId int [ref: - Person.Id, not null]
  TaskId int [ref: - LearningTask.Id, not null]
  Status enum
}

table FinishedModule {
  Id int [pk]
  PersonId int [ref: > Person.Id, not null]
  ModuleId int [ref: - LearningModule.Id, not null]
  FinishDate date
}

table PersonModuleStatus {
  Id int [pk]
  PersonId int [ref: - Person.Id, not null]
  ModuleId int [ref: - LearningModule.Id, not null]
  Status enum
}
```

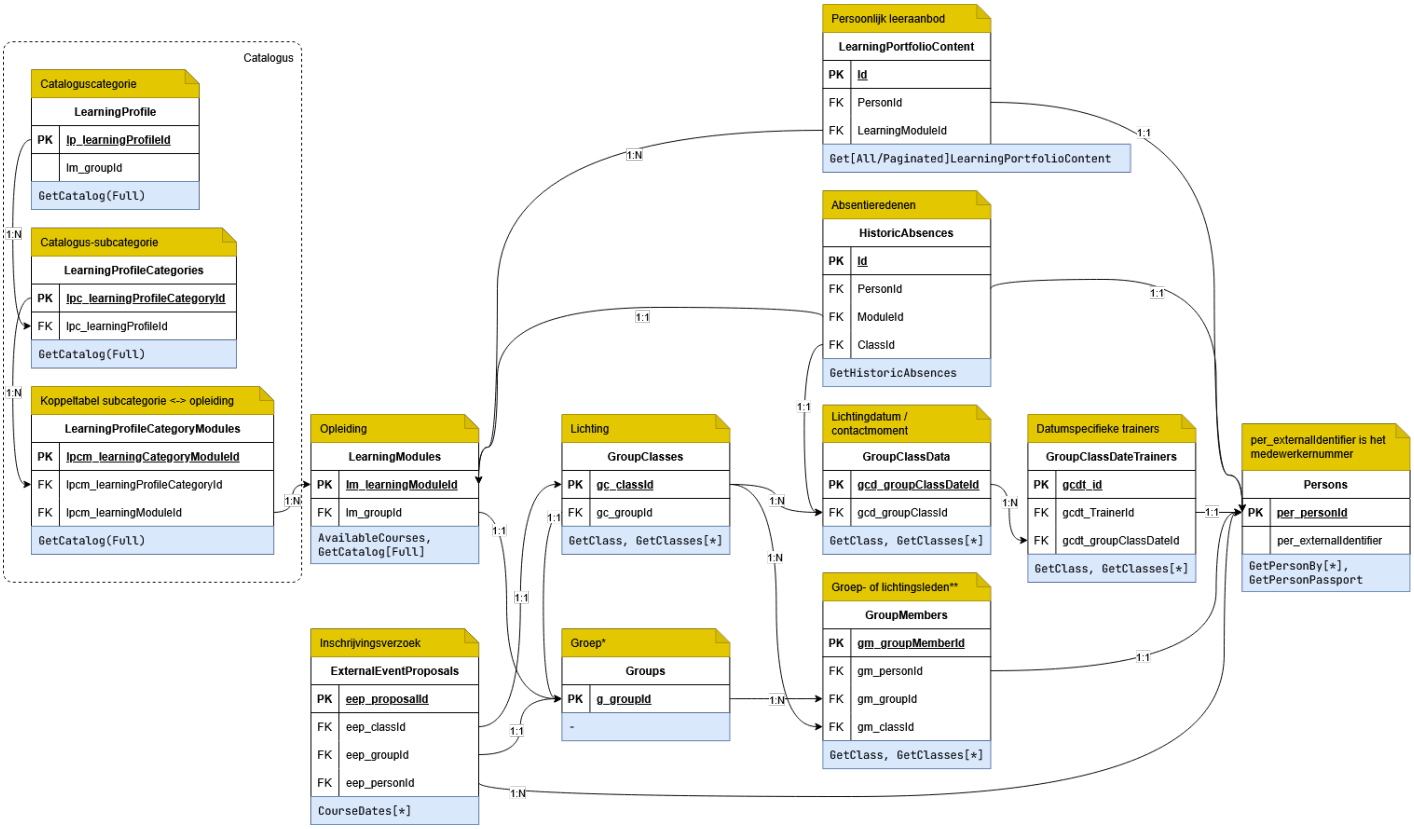
Evaluaties

> Deze pagina is nog in aanbouw.

Correspondentie API-datamodel en Pynter-datamodel

In onderstaand diagram staat voor de belangrijkste API-endpoints uitgelegd hoe deze correspondeert met de originele data in Pynter. In het geel staat de "menselijke" naam voor een item, in het blauw de namen van de API-endpoints waar deze data in terugkomt.

> Wanneer er in de naam van een API-endpoint een gedeelte tussen blokhaken staat, betekent dat dat hier meerdere endpoints bedoeld worden. Bijvoorbeeld: `CourseDates[*]` betekent alle API-endpoints die beginnen met `CourseDates`, zoals `CourseDatesByModules` en `CourseDatesWithStatus`.



* Groepen worden niet direct teruggegeven via de API, maar staan voor de compleetheid hiertussen omdat deze als koppeltabel tussen opleidingen en inschrijvingen (`ExternalEventProposals` / `GroupMembers`) gebruikt wordt.

** De `GroupMembers` tabel wordt voor zowel groepsleden als lichtingsleden gebruikt. Wanneer het een groepsinschrijving betreft staat `gm_classId` op 0.