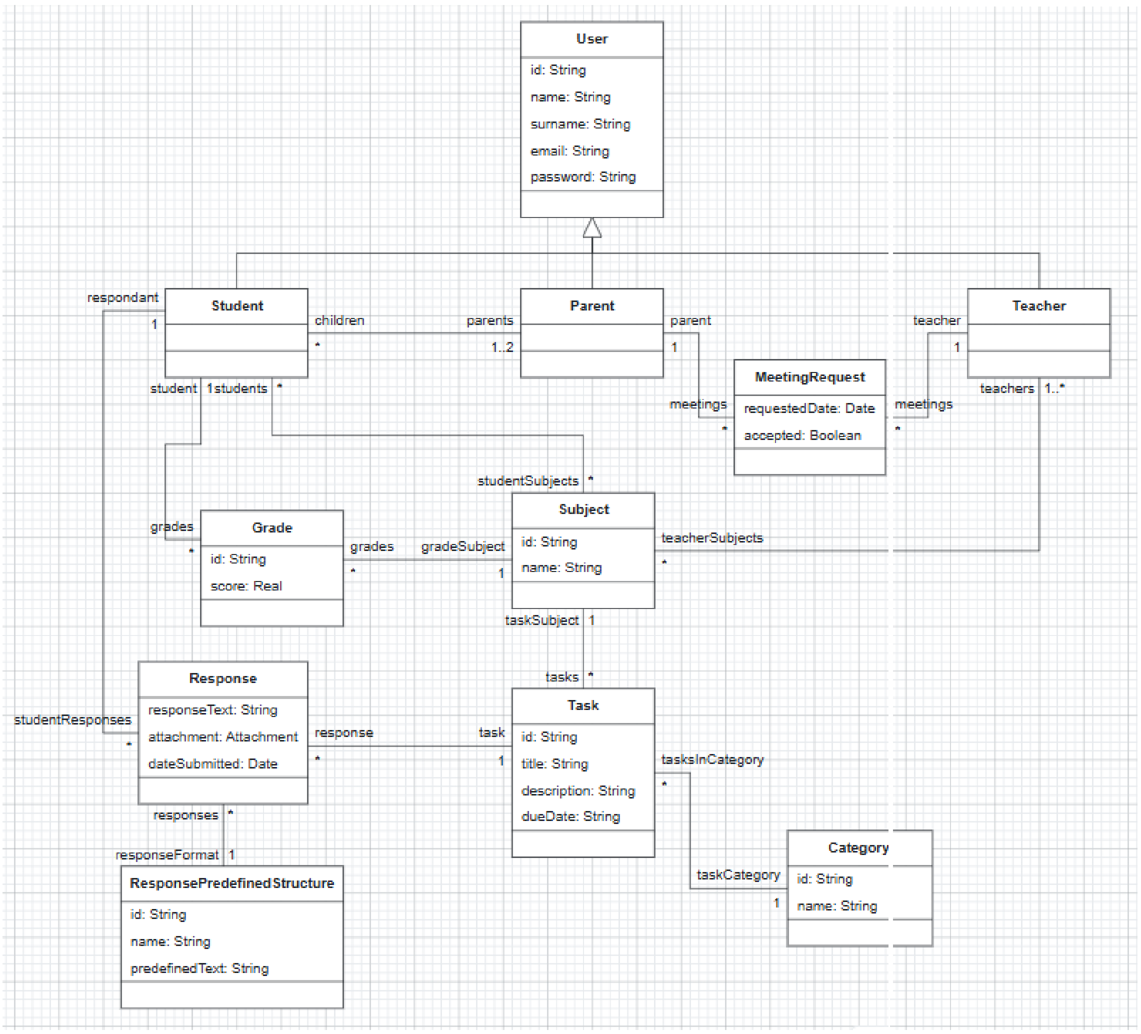
Introduction



This domain model is on a server for a simple school platform that provides students, teachers, and parents with a number of basic functionalities.

- There are subjects (Subject) that students attend (Student) and that are managed by teachers (Teacher).
- There are tasks associated with a subject. Each task is assigned to a category, and from a category all the tasks associated with it can be consulted.
- Students (Student) can respond to tasks (Task) that are associated with the subjects (Subject) in which they are enrolled. Responses are associated with a predefined response template (ResponsePredefinedStructure).
- A student can have several grades (Grade) in a single subject (Subject).
- Parents have one or more children (Student). Children have 1 or 2 parents.
- Parents and teachers can request meetings with each other (MeetingRequest).

Clients

The above model is a global model located on the server.

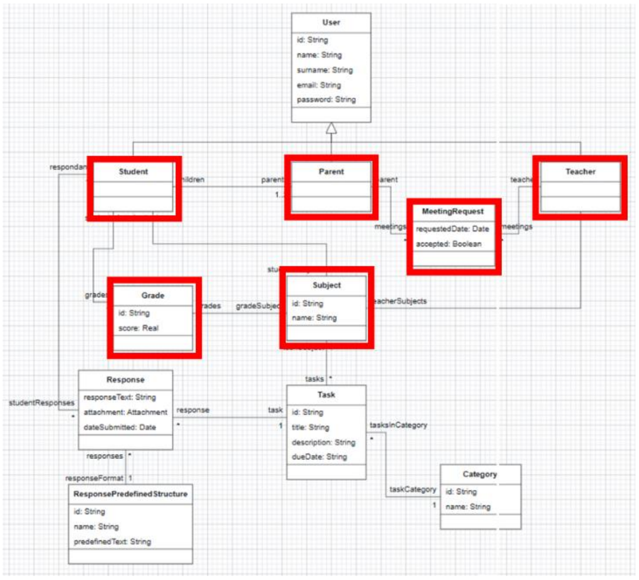
It is intended to develop several independent clients (e.g. mobile applications) for different types of users. All of these clients will communicate with the model located on the server.

For example, a ParentApp mobile application could be developed, with which parents can identify themselves and see their children's grades and subjects, and request meetings with teachers.

These clients are intended to be as independent of the server as possible and can operate offline if necessary.

For this to be possible, customers need to have their own on-premises domain model. Each client's domain model will be a subset of the server model, being a simplified and adapted version that responds to the specific needs of that client.

In the case of ParentApp, many of the classes on the server would not be relevant and would not need to be in the local model. For example, they don't need access to their children's assignments or answers, but they do need access to notes.



Because customers for different types of users will have different requirements, their domain models will be different.

Instances of client objects

Client objects can have two different sources:

- Created from scratch in the client itself.

- Obtained from the server through queries.

For example, the **ParentApp** app, used by parents, will not be able to create new instances of subjects or grades. Instead, you will have them by making queries to the server.

However, the **ParentApp** app will be able to create new objects from the MeetingRequest class, to request new meetings with teachers.

User-Accessible Information

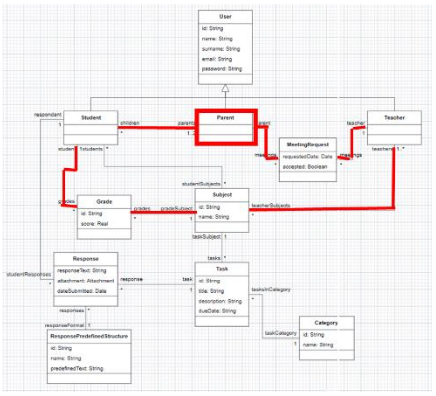
The **ParentApp** app is designed to be accessed by a single user identified in the app.

When a parent is identified, a query is made to the server to obtain from among all the instances of parents existing on the server, the one that corresponds to it.

The query will not bring only the parent's instance, but an aggregate that will also include all the other instances relevant to it, which can be accessed by browsing through their relationships (you can think of a JSON with all the relevant information for a given parent).

For example, from the Parent instance you will be able to navigate to your children's Student instances, and from each of them navigate to the grades that correspond to them, from there to their subjects, and from them to the teachers who teach them.

Even though the ParentApp domain model includes the Subject class, it is not possible to access all Subject instances on the server, only those that can be navigated from the connected parent's instance.


User
id: String
name: String
surname: String
email: String
password: String
respondant
1
Student
children
*
parents
1..*
Parent
parent
MeetingRequest
requestedDate: Date
accepted: Boolean
meetings
*
meetings
*
teacher
1
Teacher
teachers 1..*
student
1
students
*
studentSubjects
*
grades
*
Grade
id: String
score: Real
grades
*
gradeSubject
1
Subject
id: String
name: String
teacherSubjects
*
taskSubject
1
tasks
*
Response
responseText: String
attachment: Attachment
dateSubmitted: Date
studentResponses
*
response
*
task
1
Task
id: String
title: String
description: String
dueDate: String
tasksInCategory
*
responses
*
responseFormat
1
ResponsePredefinedStructure
id: String
name: String
predefinedText: String
taskCategory
1
Category
id: String
name: String