Bachelor thesis

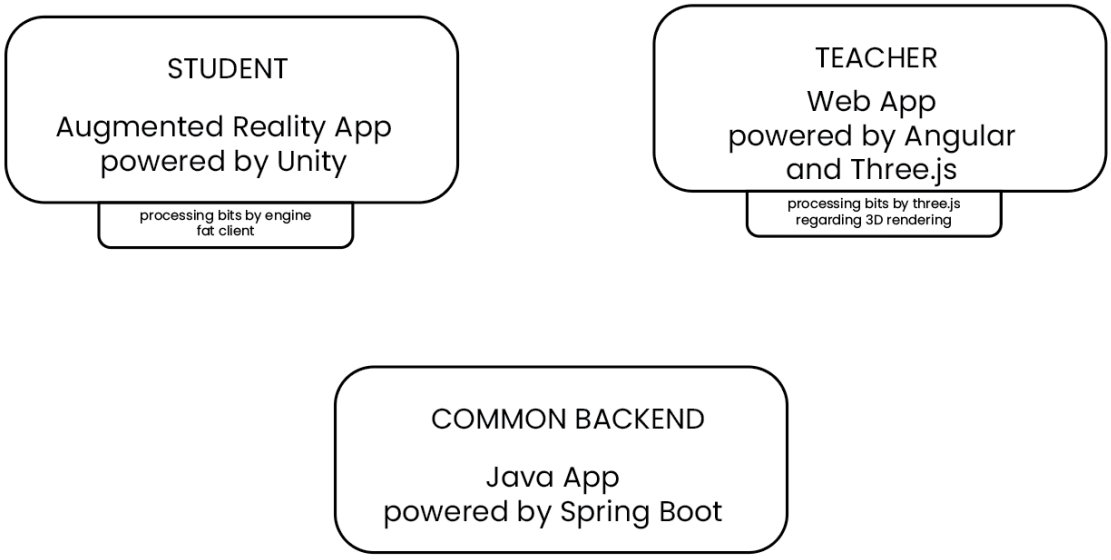
Architecture
Project diagrams

Vlad Bartolomei

Group 30643
Profesor îndrumător: Victor Ioan Bîcu

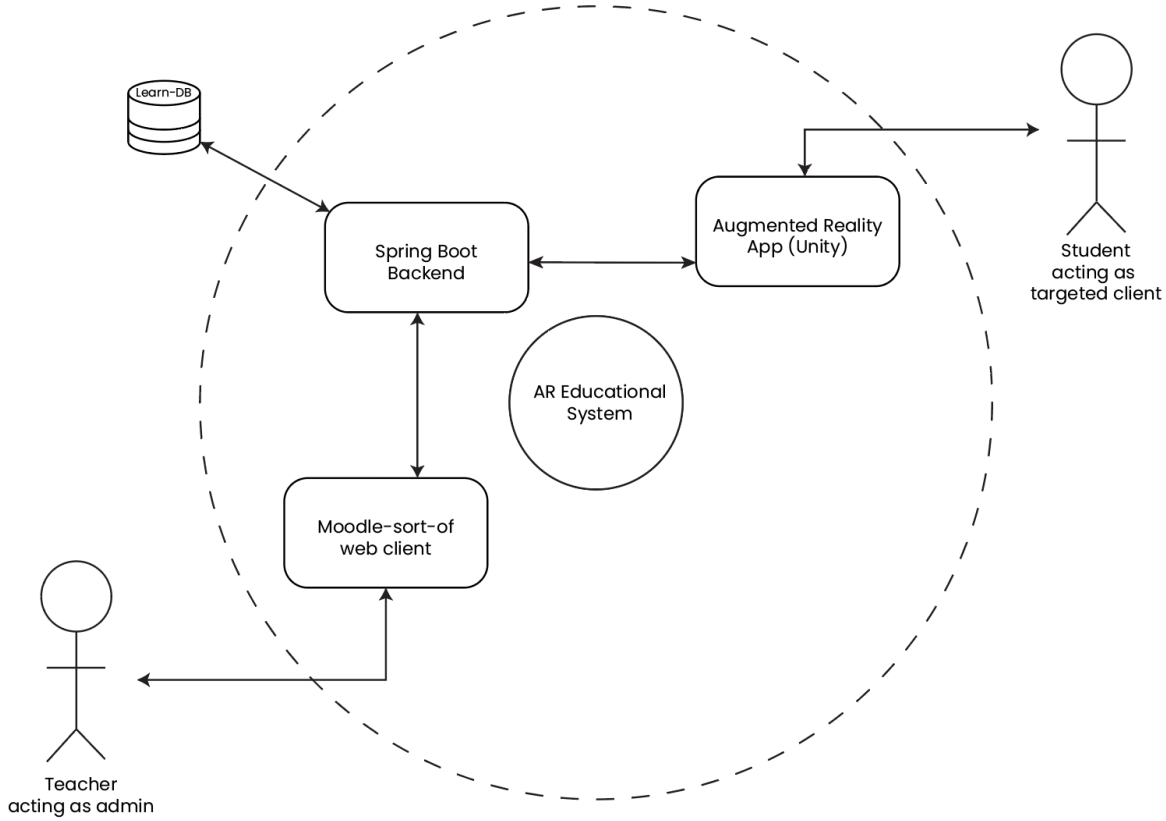
Facultatea de Automatică și Calculatoare
Universitatea Tehnică din Cluj-Napoca
România
8 Decembrie 2024

1. Domain Driven design



The app consists of two sides: student side and teacher side. They will have access to the same server, a monolith Spring Boot App which will query one big database.

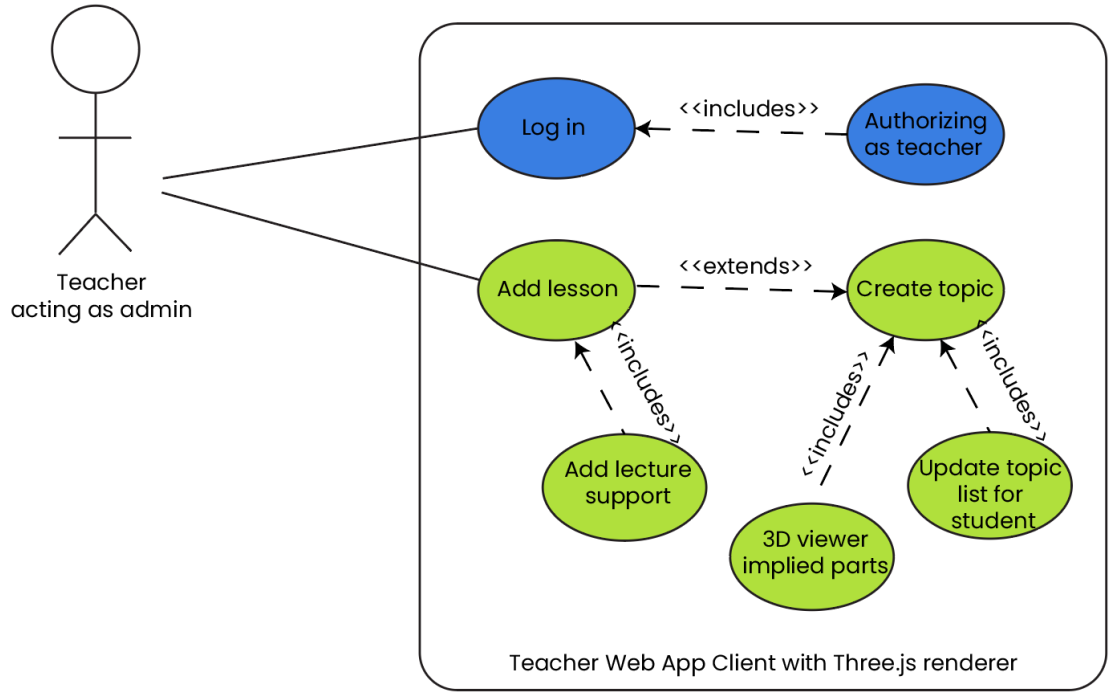
In other words:



2. Use-case diagrams

Before plunging into technicalities like how the architecture of each module (or domain if you wish) looks like, let's have a look at how shall the app be used. As you probably realised, the approach to design this project is top-down.

Use case #1: Teacher is developing a learning module (with pictures)



Primary actor: Teacher

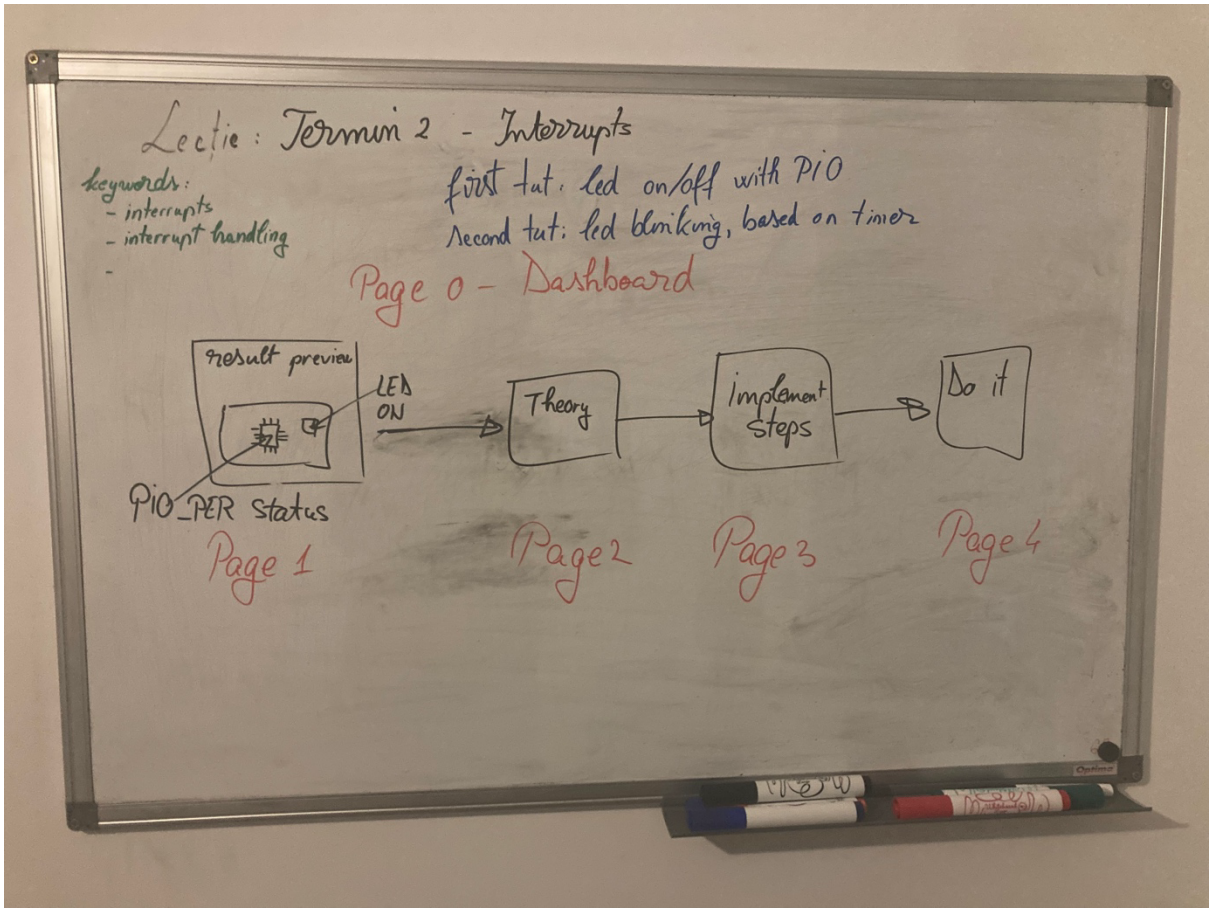
Success scenario steps:

1. Actor logs in. Account exists and it's valid => authorize actor as the teacher
2. Reroute to dashboard (common to student and admin, some extra components and features are rendered for teacher)
3. Actor accesses Add Lesson feature
    a. Based on his know-how and subject curricula, actor creates a new lesson accordingly
    b. Actor gives some basic metadata about this lesson unit: **description**, **objective**, **prerequisites** (?), **keywords**, **targeted components**.
    c. Actor is redirected to the 3D rendered, where he marks and labels target components
4. Actor saves lesson. Procedure is finished

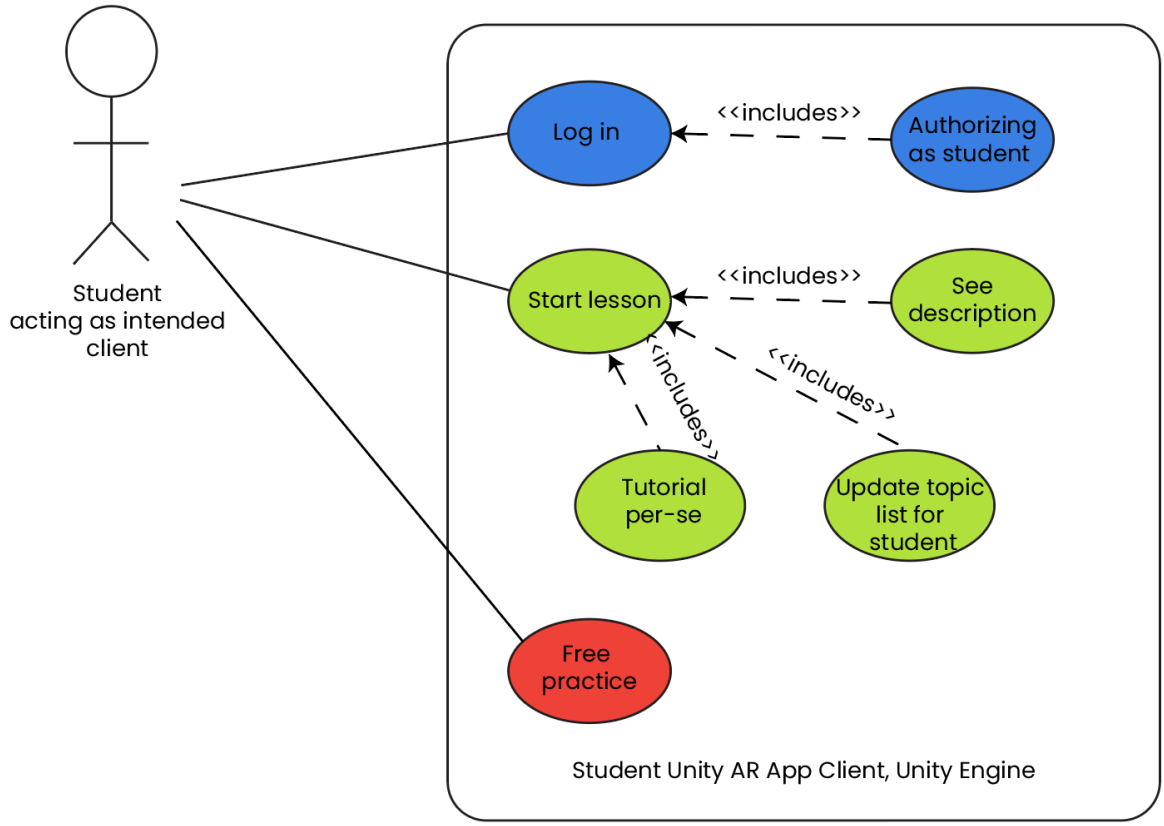
Failure scenarios:

1. If login fails => reattempt login or register with a new account
2. If actor abandons lesson or the program suddenly stops working => app has a procedural saving. At each step (step 3a, 3b and 3c) progress is saved into database. If this pipeline is not completed, the teacher will see in its dashboard this lesson as Incomplete, whereas the student won't even know about its existance.

Pipeline (step-by-step):



Use case #2: Student is following a learning module



Primary actor: Student

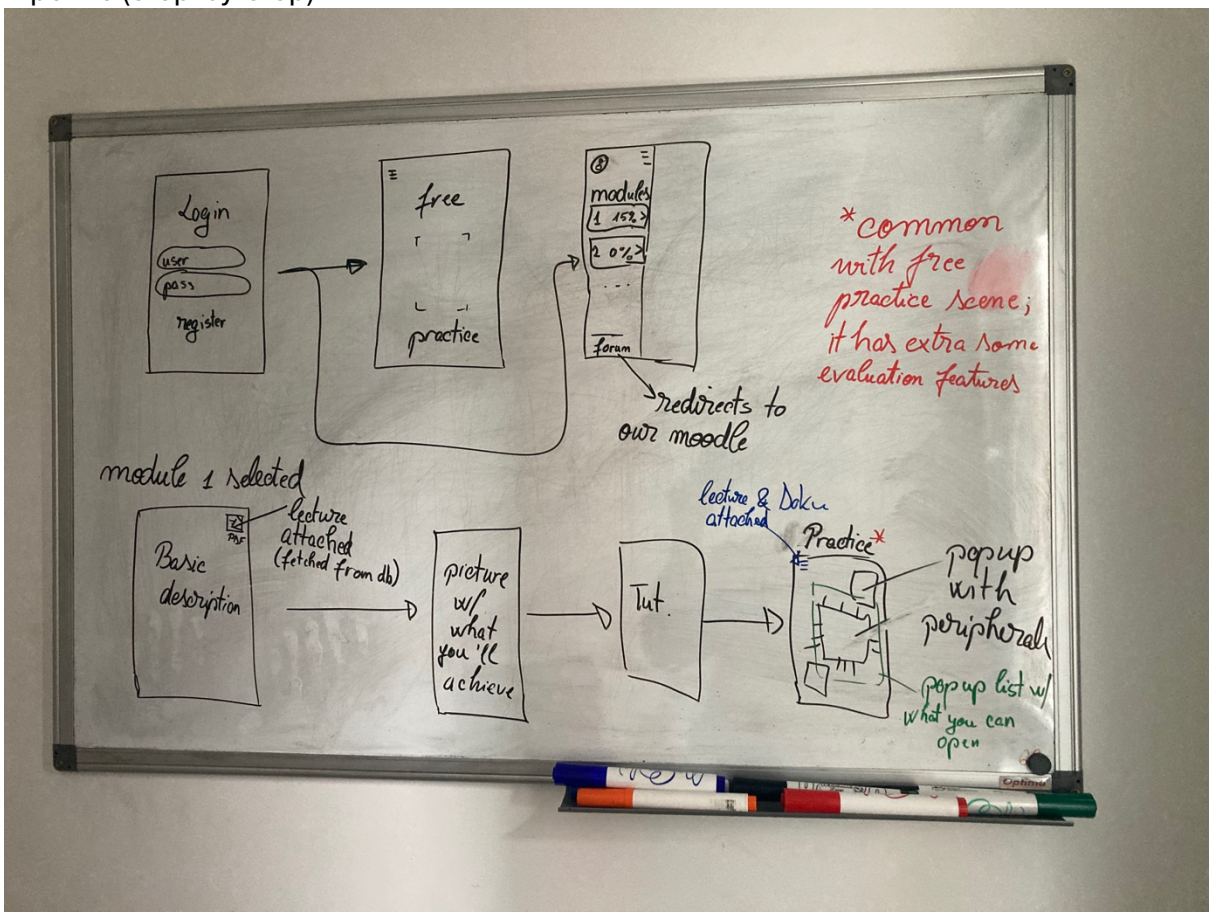
Success scenario steps:

1. Actor logs in
2. Actor is presented with a list of modules directly fetched from the server (what the teacher has prepaired for him)
3. Actor reads that basic metadata
4. Actor is required to
    a. Detect the arduino board
    b. solve a problem based on what he just learned
5. Actor is graded

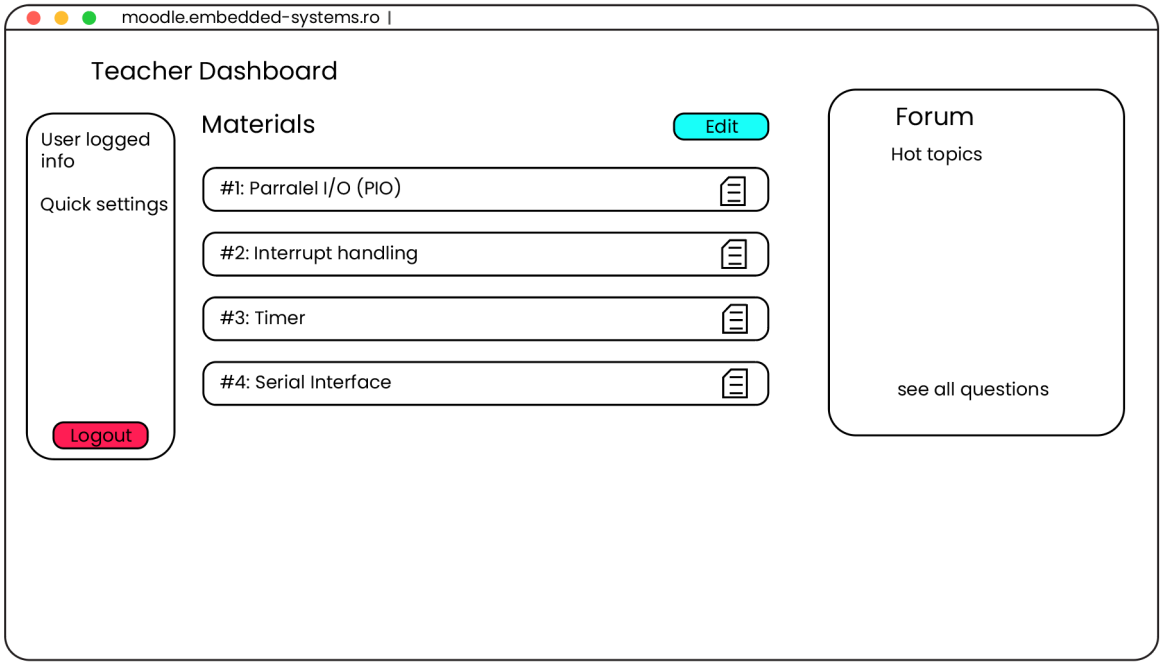
Failure scenarios

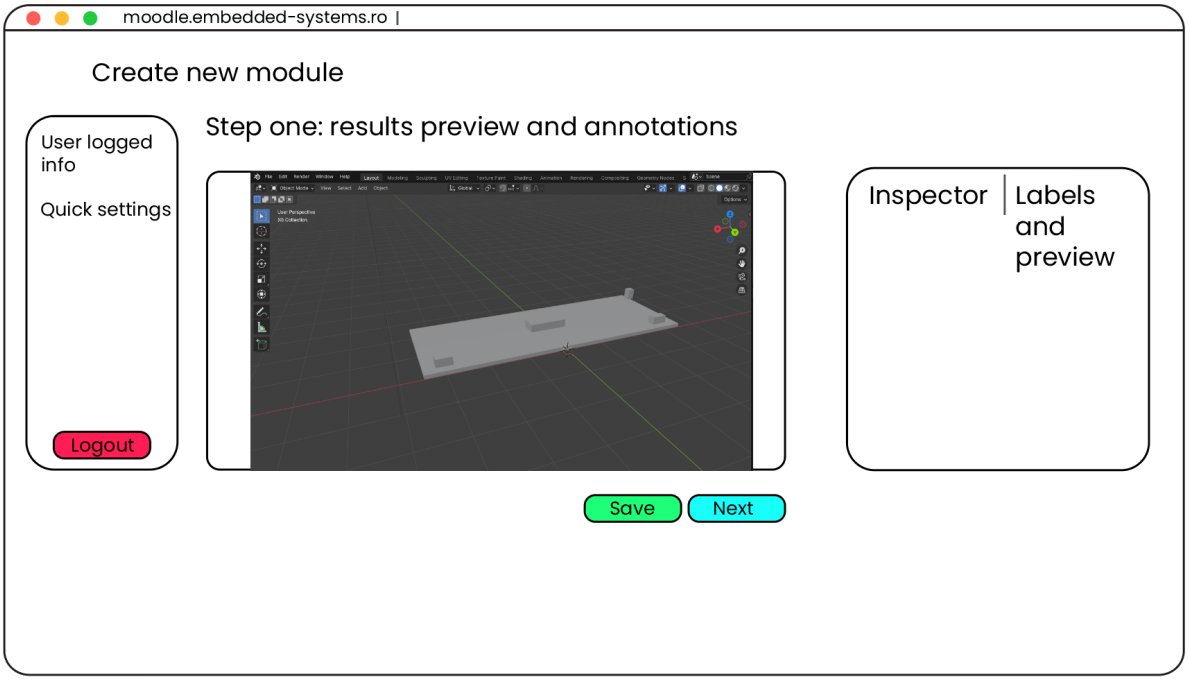
1. If login fails => reattempt login or register with a new account
2. Actor chooses a lesson to which he hasn't reached yet (we can't assume therefore that he just wants to relearn that lesson), hence he doesn't fulfil the prerequisites => that module will be greyed out
3. Actor quits while making the lesson => he will have to restart from the last completed lesson/module

Pipeline (step-by-step):

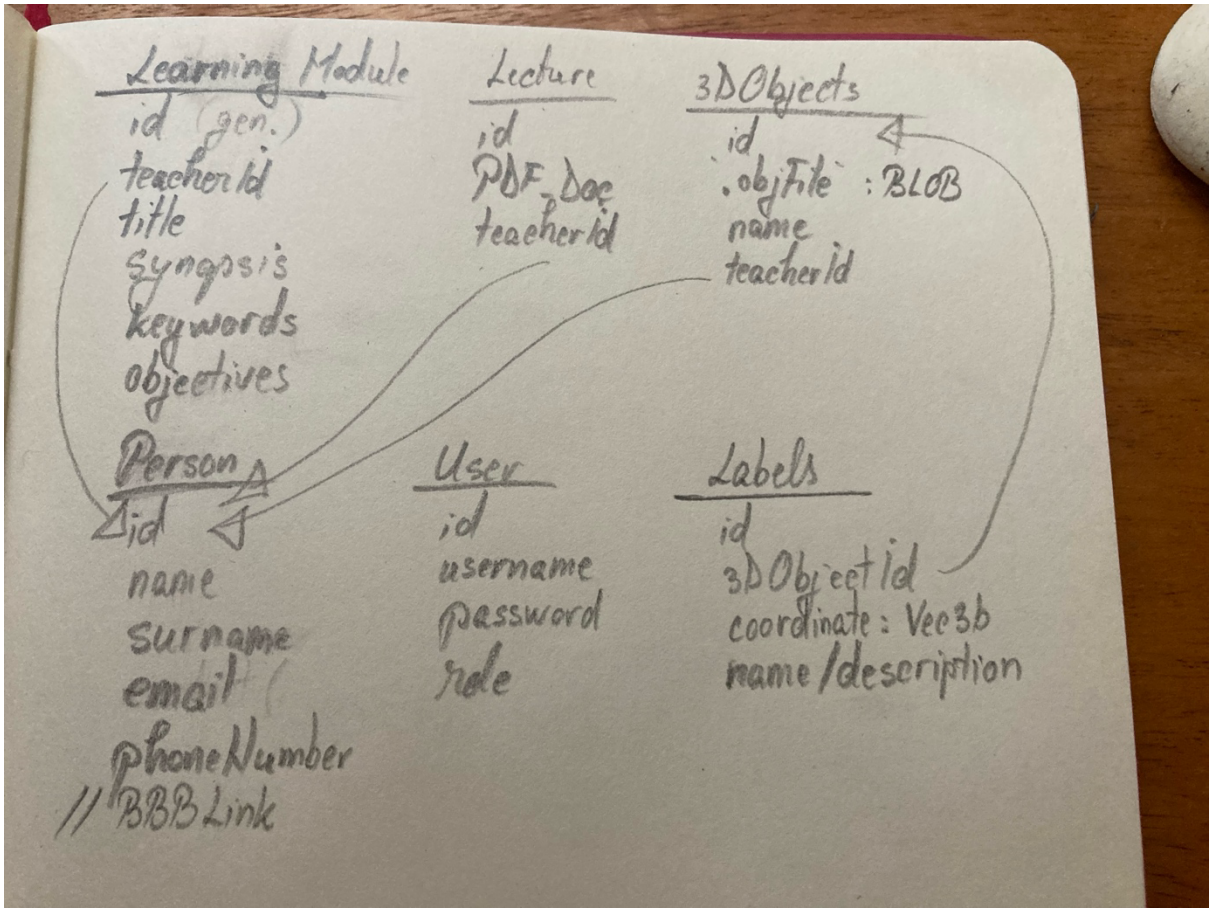


3. Entity diagrams (database)
To determine what are we going to store, we first have to understand the functionalities. As I see it, we will keep data about the users and the modules, plus other 3D objects. This is what the teacher will see:





- Lectures – linked to teacher Id
- Learning modules – linked to teacher Id
- 3D objects – linked to teacher Id
- User
- Person (either teacher or student, la grămadă)
- Progress per each learning module – linked to student Id

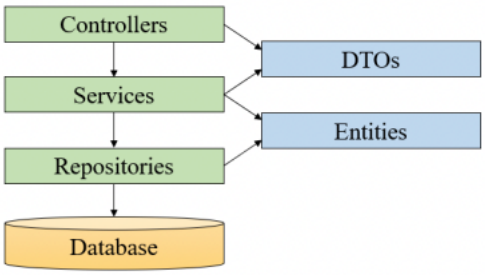


4. Other diagrams

- regarding teacher side, this is the proposed conceptual (generic architecture):

2. Project Conceptual Architecture

The conceptual architecture of the system is presented below.



- A sequence diagram isn't at the moment doable, since we are codeless atm.