Template general

[Obiectiv general]
Realizarea unei aplicații pe una dintre temele specificate, cu back-end RESTful care accesează date stocate într-o bază relațională pe baza unui API de persistenţă și date expuse de un serviciu extern și frontend SPA realizat cu un framework bazat pe componente.

[Limitări tehnologice]
- Front-end-ul trebuie realizat cu ajutorul unui framework bazat pe componente (React.js, care este acoperit în curs, sau Angular 2+, Vue.js)
- Back-end-ul trebuie să aibă o interfață REST și să fie realizat în node.js
- Stocarea se va face peste o bază relațională și accesul la baza se va face prin intermediul unui ORM
- Codul trebuie versionat într-un repository git, cu commit incrementale cu descrieri clare

[Stil și calitatea codului]
- Aplicație reală, coerentă din punct de vedere al logicii de business
- Codul trebuie să fie bine organizat, numele variabilelor trebuie să fie sugestive (și trebuie să se utilizeze un standard de numire oricare ar fi el e.g. camel case), codul trebuie să fie indentat pentru a fi ușor citibil
- Codul trebuie documentat cu comentarii la fiecare clasa, funcție etc.
- Aplicațiile care nu funcționeaza nu primesc punctaj. Se poate însă demonstra doar funcționarea back-end-ului sau a front-end-ului
- Opțional: test coverage

[Livrabile parțiale] - 3 etape (livrare se face introducând un link la un repository într-un google form; cadrul didactic coordonator va fi invitat ca un contribuitor la repository) - nelivrarea la o etapă intermediară reduce punctajul maxim cu 10% (i.e. dacă punctajul maxim este de 5 puncte din nota finală livrarea direct la final implică un punctaj maxim de 4 puncte)
- Specificații detaliate, planul de proiect, prezența unui proiect în git - 21.11.2021
- Serviciu RESTful funcțional în repository + instrucţiuni de rulare - 05.12.2021
- Aplicața completă - se livrează în ultimul seminariu (demo) - ultimul seminariu

General template

[General objective]
Developing an application on one of the specified topics, with a RESTful backend which accessed data stored in a relational database through a persistence API as well as data from an external service and an SPA front-end built with a component based framework.

[Technological constraints]
- Front-end must be build with a component based framework(React.js, which is covered in the course, or Angular 2+, Vue.js)
- Back-end must have a REST interface and must be implemented in node.js
- Storage must be over a relational database and access to the database must be done via an ORM
- Code must be versioned in a git repository with incremental commits with clear descriptions

[Code quality and style]
- Real application, coherent from a business logic standpoint
- The code must be well organized, variable names should be suggestive of their purpose and use a naming standard (e.g. camel case), the code should be indented for readability
- The code must be documented with comments for each class, function etc.
- Non working applications receive no points. However, functionality of the front-end or back-end can be demonstrated separately
- Opțional: test coverage

[Partial deliverables] - 3 stages (delivery is done by supplying a link to a repository through a google form; the coordinating university staff member will be invited as a contributor to the repository) - non delivery of an intemediary stage reduces the maximum receivable points by 10% (i.e. if the maximum grade for the project is 5 points, delivering it directly at the end means the maximum points receivable is 4)
- Detailed specifications, project plan, presence of a git project - 21.11.2021
- Functional RESTful service present in the repository + instructions to run said sevice - 05.12.2021
- Complete application - deliveded in the last tutorial (demo) - last tutorial