

# Grades

University grade management made simple

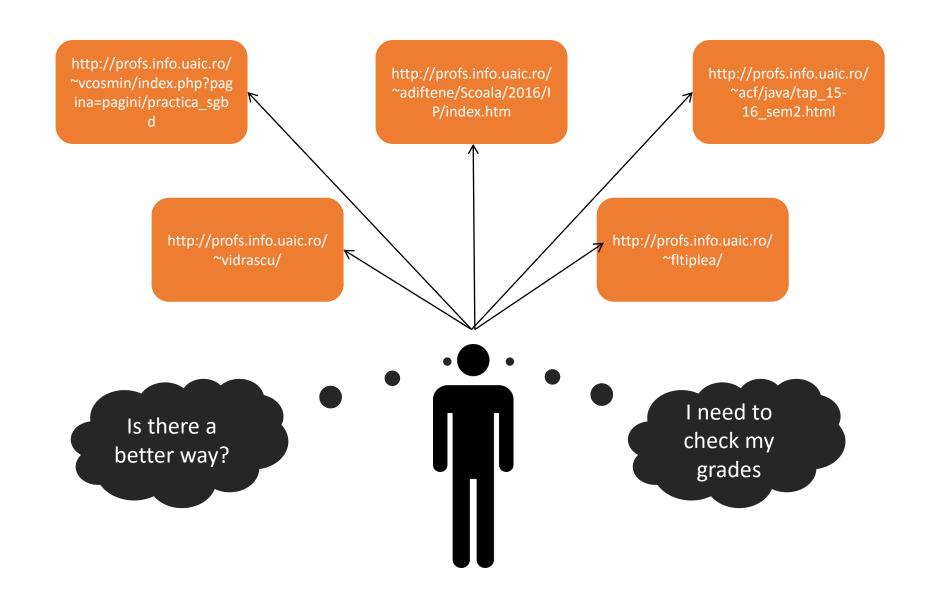
Pricop Ovidiu

Butnaru Nicolae-Adrian

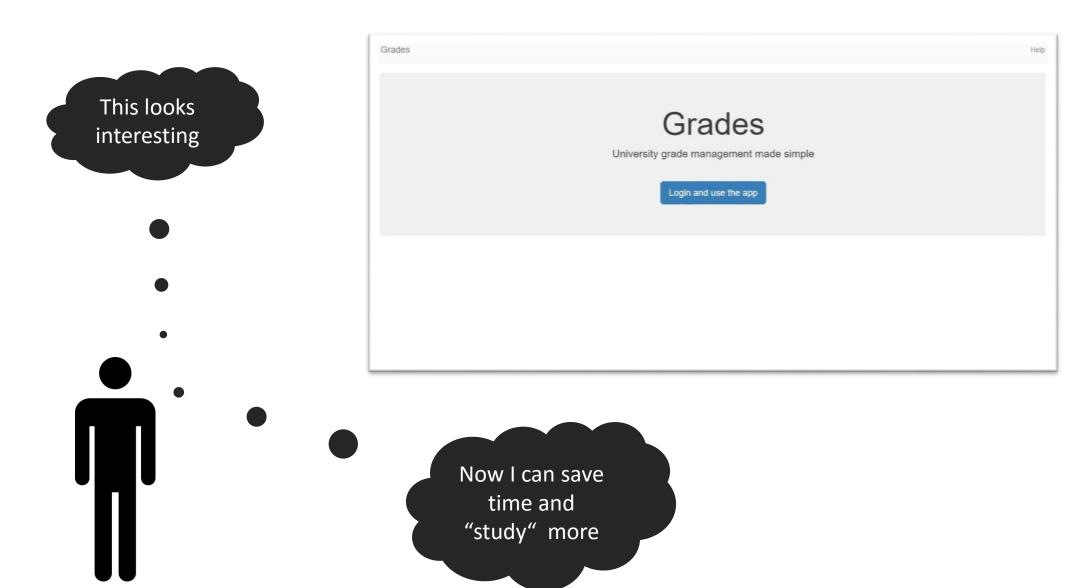
Dorneanu Cristian

Ciuc Tiberiu-Constantin

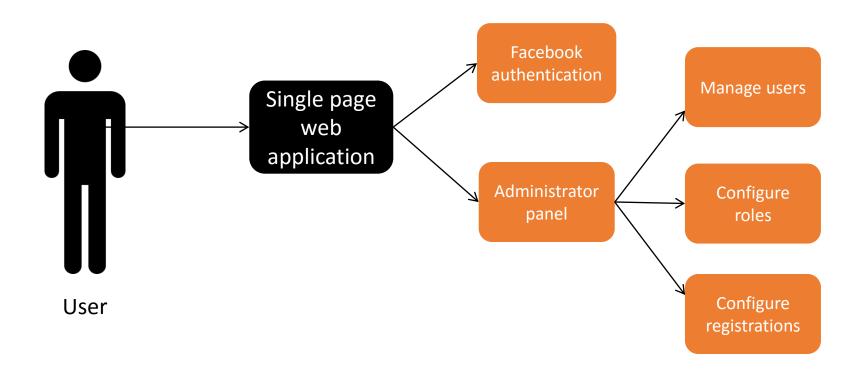
#### Context



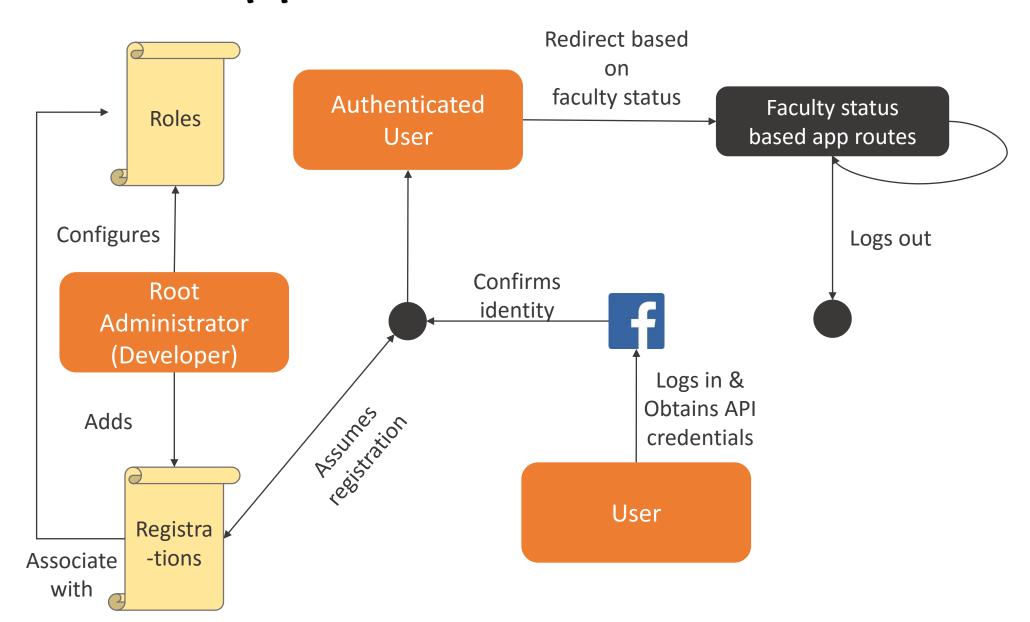
#### Intention



### Intention



## Application workflow

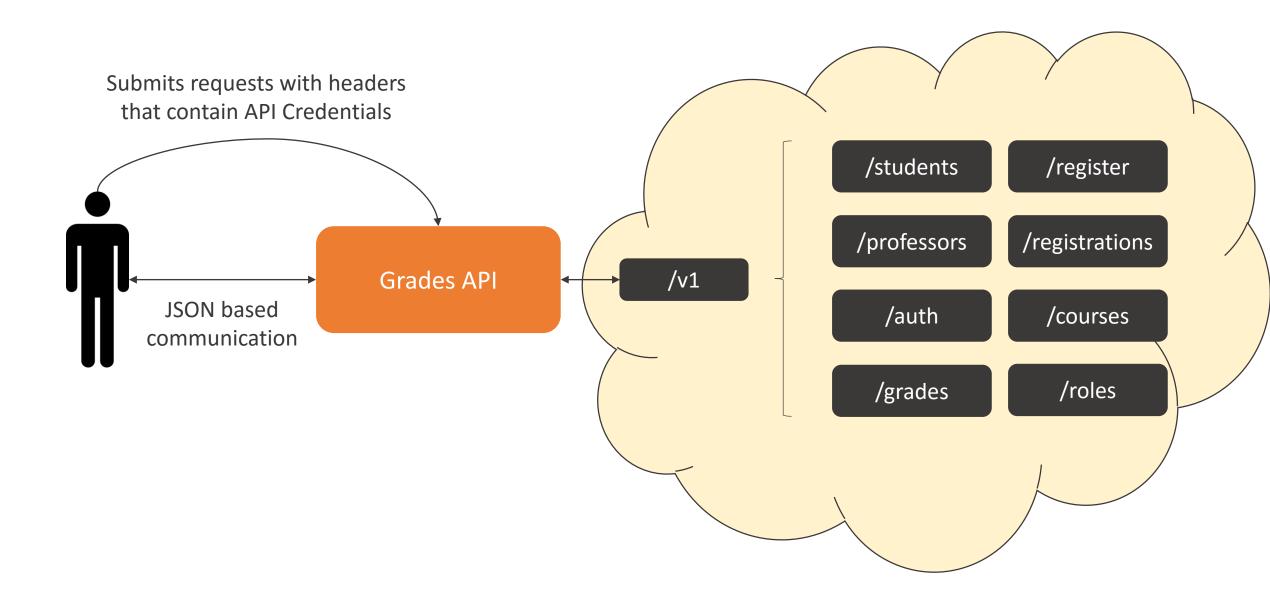


#### Authentication and Permissions

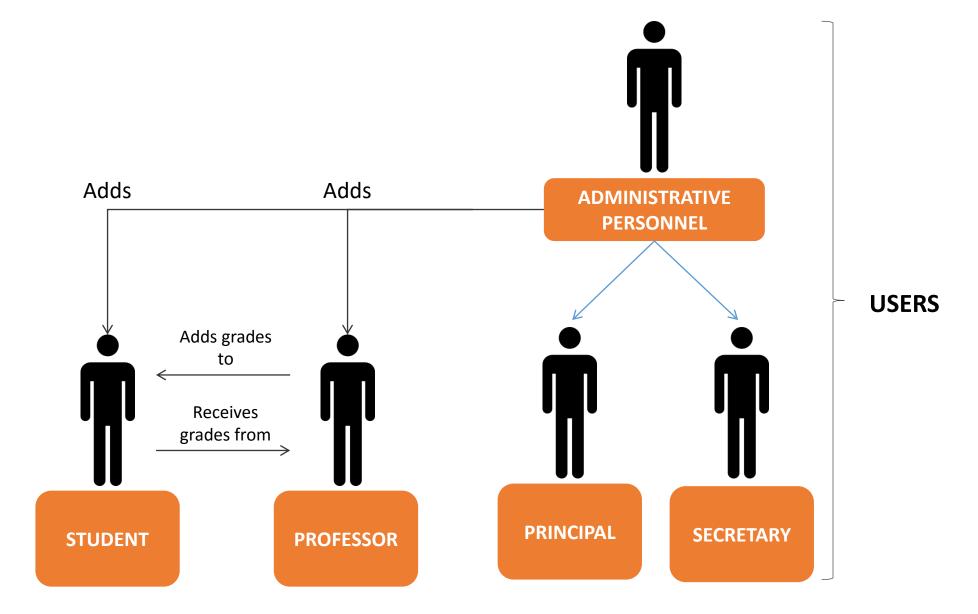
Submits request on specific resource with obtained Auth Data Credentials validation Authentication Authorization Gateway Gateway **Auth Request** Update User DB Role-based access Credentials validation (on the requested user: ..., Authorization resource) apiKey: ..., Succeeded keyExpires: ..., Actual application Redirect URL (request handler) with API Auth Data computation

Receives response (Success with data / Error)

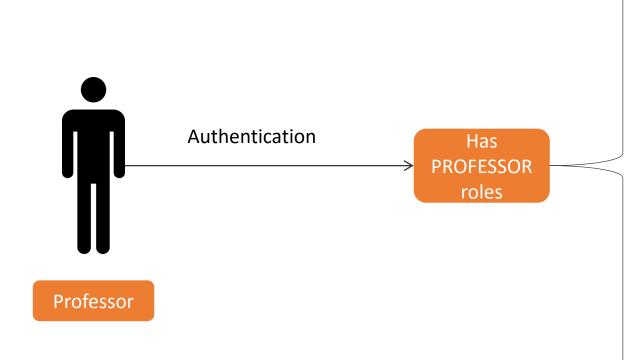
#### Grades Restful API



## Configuring Actors



#### Professors



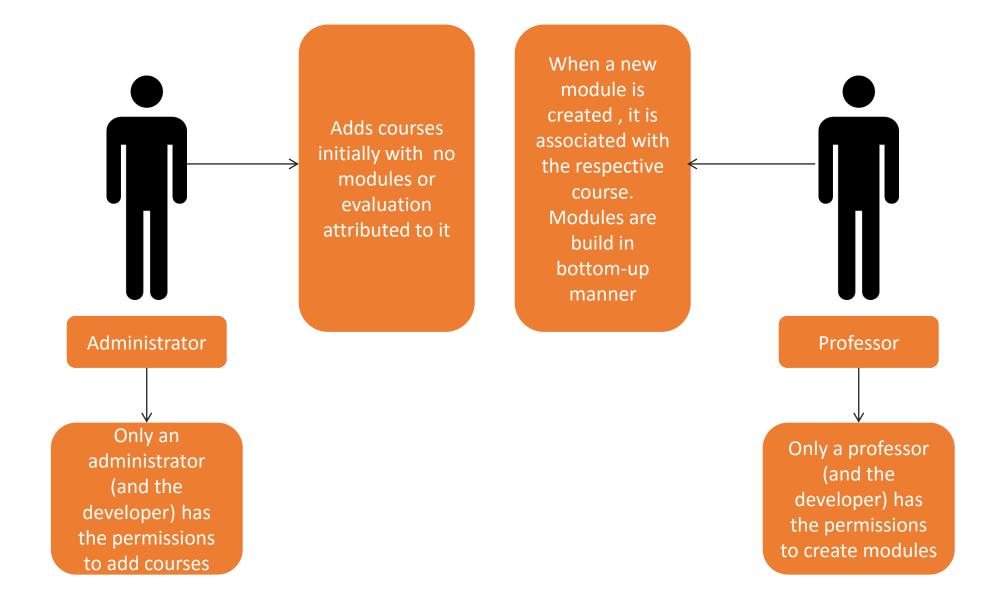
Can view all students for whose evaluation is responsible

Can add or remove modules created by him/her

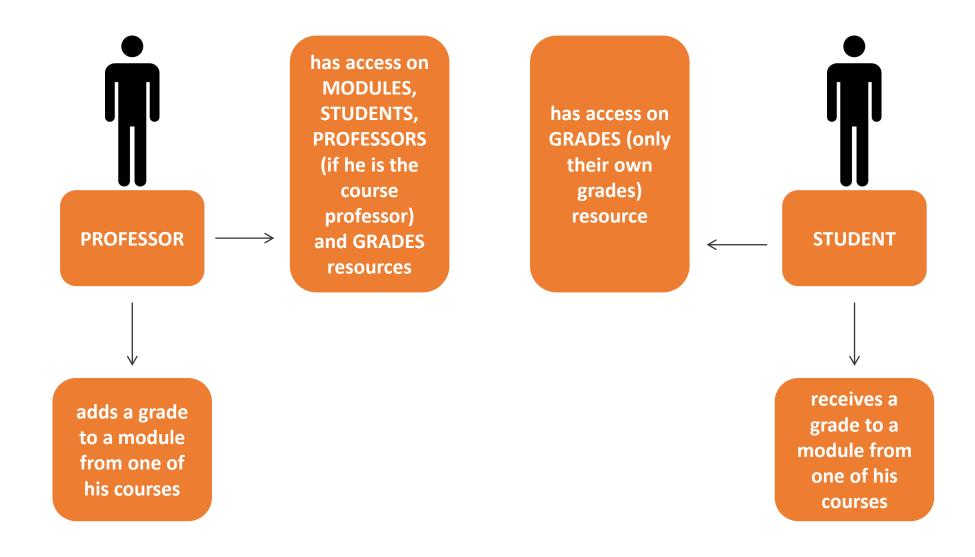
If he is a course professor
, then he can add
laboratory professors to
his course

Can add or remove a students' grades

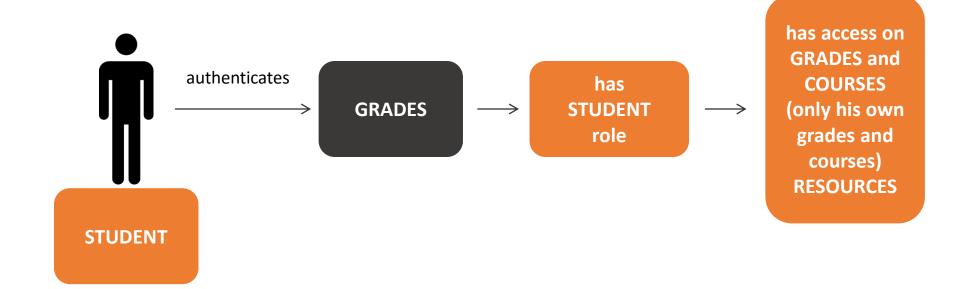
#### Courses



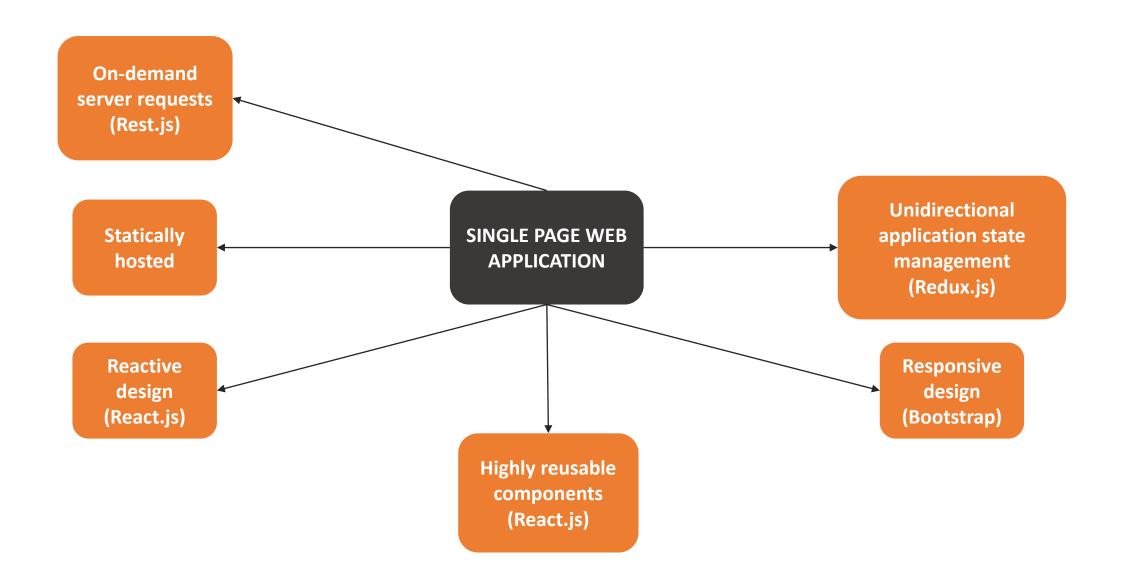
## Adding grades



### Students



#### Client & User Interface



## Architectural design keynotes

- Takes advantage of the fast Node V8 JavaScript engine
- Secure social authentication via oAuth and facebook
- Flexible and highly configurable permission system
- Scalable in terms of API action-handlers
- API can be used outside the confines of the browser
- Strongly consistent and highly flexible database system via MongoDB and Mongoose
- Multiple layers of data validation up to and including schema validators
- Clean asynchronous action handlers via async.js
- Unit tests for application-critical components

#### What we learned – Part 1

- 'Callback hell' quickly becomes a problem we found async.js as an elegant solution
- Need to be careful with Mongoose nested objects in schema
- Code is not really asynchronous until you call process.nextTick()
- Asynchronous code must be carefully crafted and not modify outer variables that are used after the async block
- Unpredictable mongoDB test results at first experience
- Github is no joke; neither is a github repository something you want to mess up

#### What we learned – Part 2

- Synchronizing code changes among team members is hard
  - highly structured (modularized) coding is necessary
- Did we mention that Github is no joke? Merging code is no joke either
- Refactoring, extracting functions so that tests still pass

# Questions



# Thank you for your time!