* Not a good idea to (ask Alan) something ab sysadmin getting the request and faculty
* Who adds nodes to the cluster? - NodeController -
* Change the repository requests - job schedule repository is better
* Job repository he thought has all of the request logic, but our thing is fine
* Subsystem for the microservice
* Sends list of all faculties until we connect - Why send the list, we can just send a test
  + The nodes don’t change and the faculties don’t change either
  + If we send the list of nodes they would be the same
  + It can just send quick tests if it’s up - pinging
* Postman has a list of request we can use instead of manually inputting
* Everytime the microservice is up we store notification until user retrieves it
  + What are the notifications for - simulate that we “send an email”
  + Doesn’t reason between getNotifications - don’t have to send the same notification again
    - If we had frontend it will arrive to user which is fine, but we also have getNotifications which is vague
    - It’s fine, our explanation is weird
    - The notification should point outwards and one connection to user
* Strategies
* Builders - jobs over 10 parameters for example and they are created all the time
  + The jobs from the requests - fine
* Two interfaces and five implementations = different ways of scheduling a job
  + Different feature to assign nodes - random, and least total resources now
    - Did the assignment specify? - “user can contribute to cluster on their own”
* Strategies in scheduling
  + Earliest possible date - fine
  + Can it ever be scheduled - rejects if it’s invalid otherwise goes to earliest
  + Latest acceptable date - from earliest completion backwards to earliest possible and defaults
  + Least busy day assignment
* When we explain these we need to be concise because we need to have equal distribution in the explanation of the strategies
* We have to use a pattern in one place in the program
* At least two strategies - we have more than that already
* Should not use chain authentication because it’s already there - it’s fine
* Avoid singletons
* Main script works at midnight for the TA - helps review our stuff
* We have folders like .idea that should not be there - put it in .gitignore
* Don’t need example-microservice and instructions
  + Clean everything
* Commits like “quick fix” should not be there - specify on what exactly
  + Make it cleaner as well
* We should make our repository professional and clean looking
* Close spring when merging
* The token verification is done on every microservice - okay
  + Will make merge request as well
* Merge requests with a lot of comments
  + too much interaction, meaning it’s too big
  + make one feature a merge and it should require less comments
  + We did way too much for one merge
* Good job on the comments
* Fast iterations with small functionalities
* Assignment is a big part of the grade, make them good and that will boost our grade more
* We are back on track now
* Spring Retrospective - do by tasks
  + “User must be able to do this… “
  + Product owner and sprint managers work on these in professional settings
* Mutation or boundary testing - there is a choice
  + We can send him a message when we choose to make sure