For Oblig 4 we have implemented Cassandra db implementation, this means that we have set up a Cassandra Db, that we run through cassandras docker file. Cassandra Db Is a nosql database that is made for scalability and availability as its main objective.

For Cassandra db we have made separate tables for each of our previous data classes(patient, symptom, department and doctor) in the project these files are named PatientCass, symptomCass and so on. These set up the tables in cassandraDb and have their previous id's as primary keys.

Each of these new tables have repositories setup to use CassandraDB's methods, such as find, findall, findallbyid and so on, and some have their own methods implemented. Which allows us to save, and receive data to and from these tables. The naming scheme of these Repositories on the form of the normal repository with Cass after them. Eg. PatientRepositoryCass.

To intialise these tables we have set up a codelinerunner which saves some data in these tables to make it easy to test the methods for getting out data from the api.

In the codelinerunner you will find that we have opted to clear the data from Cassandra db on startup of the program. In a final deployment this would not be optimal, but this is an easy way to be able to run several instances of the app and not create the same tables on startup. This setup would have to be changed for a full deployment, as of course deleting data for each new server instance would be very detrimental.

You will also find, as we have noted in the readme that we only have a replication-factor of one. This has to do with us only setting up one node of Cassandra-db, and therefore there is no other nodes to replicate on. This could be implroved upon by for example setting up a Kubernetes cluster, with several cassandraDb nodes, so that data could be replicated.

The replication of nodes is highly important to both the availability of the api(as one node could go down and the api would still be available), as well as for the data security. If your one single node goes down, that data is lost forever, and thus the Recommended way of setting it up is to have nodes on different servers, on different locations so that the whole service will not be effected by a location losing power, going down another way. This is the main point of a distributed system.

Even though we do not have a distributed system at this time(as previously discussed), it could easily with the previously mentioned methods be made distributed.

To conclude the part of our database implementation, it does not yet have high availability, but is set up to be scalable, which in turn will provide availability.

However, as you will find in the code, the database tables are not fully implemented, and so not all the calls will return data from the database, and not all the post methods will insert it into the database. This is a flaw that stems from us not changing all the old repositories to use the Cassandra repositories classes

A Quick note on security

We were sadly not able to implement security for Oblig 4, this could have been implemented by for example setting up a Authorization server, which sole function is to authenticate the user, and provide the client side with a JWT token, which in turn will be used for the server side to see which resources the user should be allowed to access.