За допомогою mongodb хотілось завантажувати данні щоб потім їх перевикористовувати, а не постійно створювати ті самі данні, також хотілось попрацювати з нереляційною базою данних і так як багато було позитивних відгуків про mongodb я вирішив використати саме її. Ну і трохі покористувавшись нею я подумав що було б класно всю логіку соцмережі побудувати на основі бази данних.

В Mongodb доволі зрозуміла документація якщо не доходити до моменту з replica set, change streams і oplogs. Встановлення replica set яке описане в документації, або не зовсім правильно описано або я щось не правильно робив тому що в мене не спрацювали ті команди описані в їхній документації, але на різних сайтах можна знайти інформацію яка допоможе встановити.

  Встановлення mongodb з replica set зайняло в мене дуже багато часу так як я не міг розібратись як воно має працювати, які команди прописувати щоб воно в мене спрацювало, але загалом все звелось до однієї дуже хорошої статті на medium де було добре розписано що треба робити.

Загалом було зрозуміло як використовувати ті чи інші методи, але були проблеми типу доступу до певного піддокумента чи елемента масиву, також на скільки я зрозумів там не можна доступатись по індексу до елемента масиву в документі, але можна піти іншим шляхом і використати slice - ця функція вертає масив після вказаного індекса з вказаним розміром(в данному випадку можна було вказати розмір 1 і отримати елемент по потрібному індексу).

Також так як в модулі була ще задача протестить програму на певній кількості юзерів, я вирішив розбити юзерів на різні потоки і слухати базу данних і виводити якісь повідомлення якщо дії інших юзерів були повязані з тим юзером(якщо хтось лайкнув повідомлення юзера, друг щось запостив або прокоментували пост юзера). Для того щоб це реалізувати потрібно було якось слухати базу данних на зміни і тут було два варіанти через oplogs або через change streams, і потрібно використати change streams бо oplogs застарівший варіант, але так як спочатку недочитав документацію я почав гуглити oplog, про якого навіть було декілька статтей на хабрі але якось воно не зовсім робило те що мені потрібно, було дуже складно розписано і тоді я в якийсь момент всежтаки натрапив на change streams і використав саме це.

Для того щоб використати change streams потрібно було встановити replica set. От тут і почалась найскладніша частина тому що майже нічого не зрозуміло як воно має працювати і чого команди з документації не спрацьовують видаючи різні помилки типу “цей localhost ви не можете використати бо щось там вже використовується спробуйте інший” або якісь ще інші але я вже не памятаю. Після встановлення replica set треба було розібратись як працюють ці change streams в scala mongodb driver. Ну як і загалом mongodb scala driver вони використовують observables. Тут треба було більш детально з цим розібраться бо оказалось не все так просто як було раніше з використанням mongodb. З’явились нові поняття потипу синхронізації потоків - в офіційному гітхабі mongodb scala driver був один приклад де використовувалось Countdownlatch. Тут і implementing observer trait(інтерфейсу observer) з різними методами які починаєш розуміти тільки після того як прочитав більш детально про observables і використання resume tokens які я не зрозумів для чого використовувались, але я обійшовся і без них на даному етапі. Також якщо немає інформації по використанні на scala то є сенс гуглити і приклади на java, навіть один приклад який реалізований і на java і на scala, може бути більш зрозумілий на першій мові а ніж на тій що треба, а потім з розумінням що треба зробити можна і використати трошки поіншому але вже на scala.

Загалом дуже цікаво було, дуже багато корисної інформації і здаєтсья це хороший варіант для вступу-знайомства з асинхронним програмуванням та використанням nosql db на scala або java. Але є дуже багато складного матеріалу який треба далі вивчати і з ним розбиратись.

Також до мінусів mongodb scala driver - ОДИН приклад на change streams на оф гітхабі і з дуже маленькою кількістю інформації і так як я не дуже обізнаний в мене виникало дуже багато питань і не на всі я знайшов відповіді.

До плюсів: з основними функціями потипу вставки елментів, пошуку і тд, було легше розібратись ніж в sqlite3 яку я використовував в минулому році в одній з лабораторних. І впринципі якщо добре читати документацію то багато помилок можна уникнути і може навіть менше витратити часу.

Ще я використав Circe parser для декодінга json документів. Там впринципі все було зрозуміло, дуже легка для розуміння бібліотека(принаймні мені не треба було багато функцій і всі вони були прості) і дуже спрощує переведення json в обєкти класів.

Посилання:

<https://docs.mongodb.com/manual/crud/>

<https://docs.mongodb.com/manual/changeStreams/>

<https://docs.mongodb.com/manual/replication/>

<http://reactivex.io/documentation/observable.html>

<http://mongodb.github.io/mongo-scala-driver/2.6/getting-started/>

[https://github.com/mongodb/mongo-scala-driver](https://github.com/mongodb/mongo-scala-driver/blob/master/driver/src/it/scala/org/mongodb/scala/DocumentationChangeStreamExampleSpec.scala)

<https://medium.com/@katopz/minimal-mongodb-replica-set-osx-76dc9dc36018>

<https://circe.github.io/circe/parsing.html>

<https://circe.github.io/circe/codecs/adt.html>

<http://mongodb.github.io/mongo-scala-driver/2.6/scaladoc/>

<https://mongodb.github.io/mongo-java-driver/3.6/driver/tutorials/change-streams/>

<https://habr.com/ru/post/277669/>

<https://www.programcreek.com/scala/rx.lang.scala.Observable>