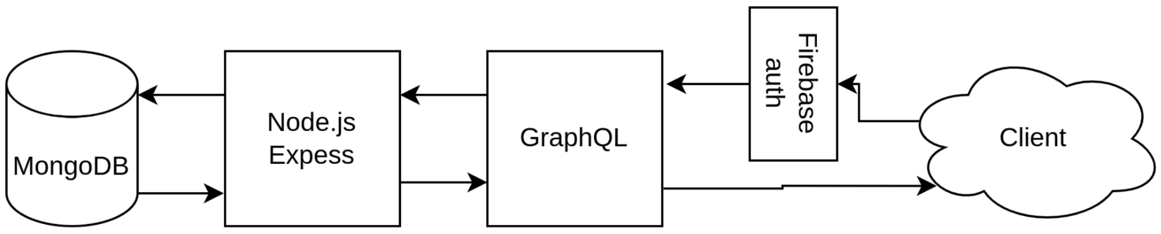
# Lending app backend

(Quick Overview)

The goal is to create a simple but efficient architecture that fulfils basic needs and does not create lock-in.

To achieve this, the architecture will be separated into replaceable parts.



## Database

To keep things simple MongoDB will be used.

Since we should be able to store most of the data from a single user into one MongoDB document and not exceed the 16MB size limit. This will be a simple and effective way to store data.

To ensure the 16MB limit on the document is not reached (in a reasonable time frame) we must only separate transactions into separate collections that are well indexed.

In this way, only one simple database query has to be made to fetch all essential user data. After that paginated queries can be made on the transactions collection to ensure fast loading times.

## Node.js + TypeScript (API)

There are only three objectives that must be fulfilled:

1. Get data from MongoDB to the client and provide a simple auth guard on requests.
2. Receive data from client, check+format data and store it into MongoDB
3. Precalculate data from transactions and provide a cache to reduce response time.

## GraphQL

Why GraphQL?

1. It is ideal for letting the client pick what data it needs since all user data will be stored in a single document.
2. GraphQL will also provide documentation.
3. It works great with MongoDB (JSON).

## Firebase

Firebase will provide authentication service (it allows export and migration) and great analytics.

## Node.js (Interest calculator)

This separate service will take care of adding interest to loans.

We will see how well it will work on scale.
*Maybe it would be better to calculate interest on the client side.