Persistent Queries

Why?

- Securing the endpoints
- Caching
- Less data to send when fetching queries

Securing Endpoints

- Depth Limiting
- Rate/Query Cost Limiting
- Persistent Queries (whitelisting)
- Amount Limiting of variables

```
npm install --save persistgraphql
```

```
addPersistedQueries(
  networkInterface: NetworkInterface,
  queryMap: OutputMap
)
```

```
const queryMap = require("../extracted_queries.json");
const { invert } = require("lodash");

app.use("/graphql", (req, _resp, next) => {
  if (config.persistedQueries) {
    const invertedMap = invert(queryMap);
    req.body.query = invertedMap[req.body.id];
  }
  next();
});
```

Exercise

Implement depth limit (and test it with 1) to verify it's working.

Update restaurants query to accept a limit (first) parameter. Use graphql-input-number to set min & max values.

https://blog.apollographql.com/securing-your-graphql-api-from-malicious-queries-16130a324a6b

https://github.com/stems/graphql-depth-limit
https://github.com/joonhocho/graphql-input-number
https://github.com/pa-bru/graphql-cost-analysis
https://github.com/slicknode/graphql-query-complexity

Exercise

Implement persistent queries for our backend based on the provided client application.