Period 4: Learning goals

MongoDB indexes and Geo-features

Explain about indexes in MongoDB, how to create them, and *demonstrate* how you have used them.

Indexes are special data structures [1] that store a small portion of the collection's data set in an easy to traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field. The ordering of the index entries supports efficient equality matches and range-based query operations.

```
db.collection.createIndex(keys, options, commitQuorum)
```

←SKRIV HVORDAN VI HAR BRUGT DEM →

Explain, *using your own code* examples, how you have used some of MongoDB's "special" indexes like ***TTL*** and ***2dsphere*** *and perhaps also the* ***Unique*** *Index*.

← FIND EKSEMPEL PÅ DET HER →

Explain and demonstrate basic Geo-JSON, involving as a minimum, Points and Polygons

```
export interface Point {
    type: string;
    coordinates: number[];
  }

  export interface Polygon {
    type: string;
    coordinates: number[][][];
  }

const gameArea = {
  type: "Polygon",
  coordinates: [
    [
      [12.547845840454102, 55.78666077505376],
      [12.547330856323242, 55.77908313513154],
      [12.561235427856445, 55.7763799260891],
      [12.575654983520508, 55.779372753550554],
      [12.583036422729492, 55.78511640751057],
      [12.580461502075195, 55.79370617514494],
      [12.577800750732422, 55.798724039171056],
      [12.563295364379881, 55.79819333412856],
      [12.553339004516602, 55.79409218763999],
      [12.547845840454102, 55.78666077505376],
    ],
  ],
};
```

A point is a given place on the map that contains a x and y value (maybe also z for height)

X and y = længde/ breddegrad

Polygon bruges til at beskrive større områder og kan have huller indeni.

Hullerne bruges til steder hvor der fx ikke er land eller lign.

Explain and demonstrate ways to create Geo-JSON test data

▼ Example of geo-json test data

```
const gameArea = {
  type: "Polygon",
  coordinates: [
    [
      [12.547845840454102, 55.78666077505376],
      [12.547330856323242, 55.77908313513154],
      [12.561235427856445, 55.7763799260891],
      [12.575654983520508, 55.779372753550554],
      [12.583036422729492, 55.78511640751057],
      [12.580461502075195, 55.79370617514494],
      [12.577800750732422, 55.798724039171056],
      [12.563295364379881, 55.79819333412856],
      [12.553339004516602, 55.79409218763999],
      [12.547845840454102, 55.78666077505376],
    ],
  ],
};

const players = [
  {
    type: "Feature",
    properties: {
      name: "p1-outside",
    },
    geometry: {
      type: "Point",
      coordinates: [12.549476623535156, 55.791872563546214],
    },
  },
  {
    type: "Feature",
    properties: {
      name: "p2-outside",
    },
    geometry: {
      type: "Point",
      coordinates: [12.547588348388672, 55.788060305260046],
    },
  },
  {
    type: "Feature",
    properties: { name: "p3-inside" },
    geometry: {
      type: "Point",
      coordinates: [12.558832168579102, 55.78637121080268],
    },
  },
```

```
  {
    type: "Feature",
    properties: { name: "p4-inside" },
    geometry: {
      type: "Point",
      coordinates: [12.564239501953123, 55.79201732549507],
    },
  },
];

export { gameArea, players };
//module.exports = { gameArea, players };
```

Here you see how points, gamearea and players are created

Explain the typical order of longitude and latitude used by Server-Side APIs and Client-Side APIs

- Længdebred og breddegrad
- Ingen fast konvention, find evt et udgangspunkt omkring london (fx dk) og find så ud af hvilke værdier der er hvad ud fra sund fornuft

Explain and demonstrate a GraphQL API that implements geo-features, using a relevant geo-library and plain JavaScript

▼ Schema

```
import { makeExecutableSchema } from "graphql-tools";
import { resolvers } from "./resolvers";

const typeDefs = `#graphql

type Status{
  """TRUE if coordinates was inside gameArea, otherwise FALSE"""
  status: String

  """Contains a string with a description of whether given coordinates was inside or not inside the gameArea"""
  msg: String
}

type Coordinate {
  latitude: Float!
  longitude:Float!
}

type Coordinates {
  coordinates: [[[Float]]]
}

type Point {

  """Will ALWAYS have the value Point"""
  type : String

  """Array with longitude followed by latitude [lon,lat]"""
  coordinates: [Float]
}
```

```
type Player {
  """userName of Player (or Team)"""
  name : String

  """GeoJson Point with the users location"""
  point : Point
}

"""Represents a user found, with the distance to the caller"""
type User {
  """Distance to the user searched for"""
  distance: Float

  """userName of the user found"""
  to: String
}

"""
Error for a call, with msg and statuscode
"""
type Error {
  msg: String
  statuscode : Int
}

type Query {

  """Returns a GeoJson Polygon representing the legal gameArea"""
  gameArea : Coordinates

   """Check whether caller, given his latitude and longitude, is inside the gameArea"""
   isUserInArea("Callers latitude" latitude: Float!,"Callers longitude" longitude:Float!) : Status!

   """Given callers latitude and longitude all nearby Teams will be found (inside the given radius)"""
   findNearbyPlayers(latitude: Float!, longitude: Float!,distance: Int!):[Player]!

   """
   Given callers latitude and longitude, and the userName of the Team to find, returns
   an object with the distance and the name of the user found (team)
   """
   distanceToUser("callers latitude" latitude: Float!,
                  "callers longitude" longitude: Float!,
                  "user to find" userName: String) : User
}
`;

const schema = makeExecutableSchema({ typeDefs, resolvers });

export { schema };
```

▼ Resolvers

```
import gju from "geojson-utils";
import { gameArea, players } from "./gameData";
//const { gameArea, players } = require("./gameData");

export const resolvers = {
  Query: {
    gameArea: () => {
      return gameArea;
    },
```

```
  },
};
```

Tilføj mere

Explain and demonstrate a GraphQL API that implements geo-features, using Mongodb's geospatial queries and indexes.

To do

Explain and demonstrate how you have tested the geo-related features in you start code

To do