GeoQuiz

Katharina Tunold Unstad, Lars Furu Kjelsaas, Galina Valeska Jonat

NTNU

Innholdsfortegnelse

[GeoQuiz 3](#_Toc480745010)

[Idéutgreiing: Kravspesifikasjoner og tidsplan 3](#_Toc480745011)

[Data som ble brukt 3](#_Toc480745012)

[Kravspesifikasjon 3](#_Toc480745013)

[Tidsplan satt opp i starten av prosjektet 4](#_Toc480745014)

[Rammeverket: Arkitektur og Verktøy 4](#_Toc480745015)

[Det som ble gjort 4](#_Toc480745016)

[Front-End Applikasjonen. 4](#_Toc480745017)

[Implementasjon av Log-in. 4](#_Toc480745018)

[MongoDB - oppsett*.* 5](#_Toc480745019)

[Front-End-Backend-kommunikasjon, AJAX. 5](#_Toc480745020)

[Kart. 5](#_Toc480745021)

[Point in Polygon. 5](#_Toc480745022)

GeoQuiz

Vi tenker å bygge en geoquiz som en webapplikasjon. Spillets mål er, gjennom en rebusoppgave med ulike hint, finne fram til neste destinasjon på en rute, et slags rebusløp i et interaktivt kart (inspirert av Påskelabyrinten på NRK radio).

# Idéutgreiing: Kravspesifikasjoner og tidsplan

Applikasjonen er tenkt som spill/fritidsaktivitet. Ved å gi brukeren muligheten til å se hvor han reiser hen i et kart, legges til en geografisk komponent til quizzen.

## Data som ble brukt

Vi har ikke særlig mye behov for spesielle data, annet enn selve spørsmålene som vi hentet fra boka ”Påskelabyrinten” av Viggo Valle. Polygonene vi brukte for å registrere byene er hentet fra [http://polygons.openstreetmap.fr/index.py](https://l.messenger.com/l.php?u=http%3A%2F%2Fpolygons.openstreetmap.fr%2Findex.py&h=ATP9QufWXzod_yIQnKw-Kepb1wxwe8AEmB8u9ecz3fV78RXJE0w33RSedkMdC3KVaeGZh0gQr5i000h4xTVtvYkLrM6eagUHTj3AG-SbOITVUsCQ7CD-V6-dfaYJVldl_5MhjaWF-S6iwuQyUXPKXZSZSM7OsQ). Her kan man hente ut polygonene i json format når man har funnet id til stedet/regionen en skal ha polygonet for. Opprinnelig har vi prøvd å få disse dataene fra <https://mapzen.com/data/borders/>, noe som viste seg å være litt for tungvint for formålet vårt siden dataene hadde forskjellige administrative inndelinger i hvert land.

## Kravspesifikasjon

Angående krav hadde vi følgende ønsker til funksjonaliteter i spillet:

* Følge en quiz/rebus og løse oppgavene gjennom trykk i kartet
* Muligheten til å opprette quizer
* Login-funksjon for å lagre progresjon
  + Mail som brukernavn
  + Skrive passord to ganger
  + Hashe passord for å øke sikkerhet for brukerne
* Dele sin quiz med andre f.eks via en link (vi vurderer muligheten til å skjule quizen en lager)
* Oversikt over alle quizer(bortsett fra de skjulte) på en egen side.
* For å finne riktig sted på kartet så kan en søke på steder, og kartet roterer til riktig posisjon slik at en får trykke.
* Når en har svart riktig så vil en “reise” til dette stedet/en bue viser hvor du kom fra til hvor du er nå
  + Linjer/markering i kartet som viser reiseruten din så langt
  + “Beregnet flytid er:...”
* Flerspillerfunksjon (multiplayer)

## Tidsplan satt opp i starten av prosjektet

* Ha et MVP (Minimal Viable Product) oppe innen utgangen av februar:
  + Server og nettside koblet sammen, nettside har kart
* Forberede presentasjon av prosjekt fram til den 6. mars, prøve å få til et fint UI som kan vises fram.
* Produktet er stort sett ferdig før påske (7. april), med mulighet for finpuss på fritiden i påska.
* Bruker tiden frem til innlevering på gjenværende dokumentasjon, rapport og lignende

# Rammeverket: Arkitektur og Verktøy

//TODO update arkitektur grafikk, node, angular, express, mongodb: hva er frontend/backend? Brukte jQuery AJAX til kommunikasjon (..)

Vi brukte et KANBAN-Board som vi satt opp på Trello for å holde oversikten over oppgavene som ble gjort og som gjenstår, funksjonaliteter som ønskes o.l.

# Det som ble gjort

Front-End Applikasjonen. [Include a period at the end of a run-in heading. Note that you can include consecutive paragraphs with their own headings, where appropriate.]

Implementasjon av Log-in. [Include a period at the end of a run-in heading. Note that you can include consecutive paragraphs with their own headings, where appropriate.]

MongoDB - oppsett*.* Databasen ble satt opp ved hjelp av kommandoer utført i Terminalen. Vi prøvde først å sette den opp direkte via heroku og mLab, men det viste seg at vi da ikke fikk tilgangen til databasen som vi ønsket det. Spørsmålene ble lastet opp til heroku direkte i serveren (mLab).

Front-End-Backend-kommunikasjon, AJAX. [Include a period at the end of a run-in heading. Note that you can include consecutive paragraphs with their own headings, where appropriate.]

Kart. Vi brukte leafletjs til å lage kartet siden dette var den anbefalte måten å gjøre det på. Bakgrunnskartet ble hentet ut fra mapbox. Kartet inneholder en søkefunksjon som er en OpenStreetMap Geocoder. Den bruker Nominatim- verktøyet fra OpenStreetMaps, som blant annet inneholder funksjonaliteten om å finne fram koordinatene til et sted i kartet.

Quizreisen vår starter i Trondheim. Derfor ble det plassert et marker-icon der ved hjelp av marker-funksjonen fra leaflet. Når man har gjettet riktig sted, ble det lagt til en marker på dette stedet samt tegnet en linje fram til stedet, slik at brukeren kan se hvilke steder han har reist til og i hvilken rekkefølge.

Point in Polygon. Hovedlogikken i spillet går ut på a finne ut om brukeren har trykket på riktig sted i kartet. Dette sjekkes ved hjelp av funksjonene getBounds() og contains(), begge to er innebygde funksjoner i Leaflet. GetBounds() utføres på et polygon (i vårt tilfelle polygonet til byen som er svaret på spørsmålet) og lager et rektangel rundt dette polygonet. Så sjekker contains som tar inn punktet brukeren har trykt på om dette punktet er inneholdt i dette rektangelet og returnerer true hvis dette er tilfellet. Dette gjøres lett ved å sjekke om x- og y-verdiene til det punktet man har trykket på ligger mellom minimums- og maksimumsverdiene til rektangelet.

Opprinnelig hadde vi tenkt å bruke leaflet-pip, et skript som tester om et punkt ligger i et polygon, men siden det oppsto en del problemer ved bruk av det, valgte vi å bruke getBounds() og contains().

Leaflet-Pip ville sjekket om punktet lå i polygonet ved hjelp av krysningsmetoden (intersection): Den ville tegnet en rett linje fra punktet og telt hvor ofte denne linja snittet med linjene rundt polygonet. Om det var et odd antall krysningspunkt, lå det i polygonet, et jevnt antall krysningspunkter ville vist til at punktet ligger utenfor polygonet.

Siden vi uansett ønsket å legge inn en type toleranse for klikk var det tilstrekkelig for oss å sjekke om punktet som blir trykket på ligger innenfor et rektangel, vi taper altså ikke på det.