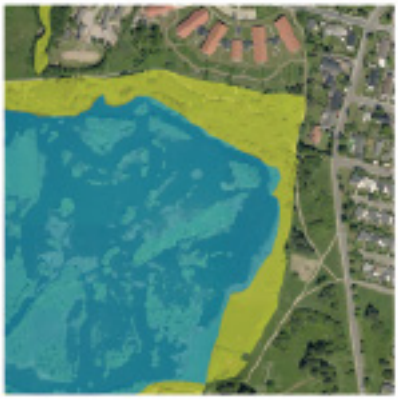
MobilGIS

Moderne webstandarder som platform for offentlig forvaltning.



Peter B. Alsing og Morten H. Fredskild, juni 2012

MobilGIS

- Moderne webstandarder som platform for offentlig forvaltning

Peter Bjørn Alsing og
Morten Heering Fredskild

Vejleder
Henning Sten Hansen
Professor
Aalborg Universitet
Inst. for Planlægning, Ballerup

AALBORG UNIVERSITET
STUDENTERRAPPORT

Titel:
MobilGIS - Moderne webstandarder som platform for offentlig forvaltning

Kandidatafhandling:
Aalborg universitet
- Geoinformation, Technology & Management

Projektperiode:
September 2011 - juni 2012

Rapporten er udarbejdet af:

.....
Peter Bjørn Alsing (stud. nr: 20101026)

.....
Morten Heering Fredskild (stud. nr: 20072991)

Oplagstal: 4

Sideantal ekskl. bilag: 115

Bilagsantal: 10

Afsluttet: d. 7 juni 2012

Synopsis

I nærværende opgave er der bliver arbejdet med udviklingen af en platformsuafhængig proof-of-concept mobil applikation, der kan benyttes som værktøj til registrering og håndtering af offentlig data. Som case studie benyttes den igangværende opdatering og ajourføring af de danske paragraf-3 områder. Implementering af projektapplikationen er primært foretaget vha. Open Source løsninger heriblandt Sencha Touch, PostGIS, GeoServer og Google Maps API.

Den udviklede webapplikation giver brugeren mulighed for at indrapportere nye paragraf-3 områder til en rumlig database, vha. en simpel touch brugergrænseflade. Brugeren har derudover bl.a. også mulighed for at foretage forskellige rumlige kald/analyser, få vist eksisterende data som overlays i applikationens kort, samt skifte mellem baggrundskort fra div. online korttjenster (Bing, Google og OpenStreetMap).

Det konkluderes, at moderne webstandarder i høj grad kan benyttes til udvikling af brugbare mobile webapplikationer i forbindelse med offentlig digital forvaltning og håndtering af rumlig data.

Rapportens indhold er frit tilgængeligt, men offentliggørelse (med kildeangivelse) må kun ske efter aftale med forfatteren

1. Forord

Dette speciale er udarbejdet ved kandidatoverbygningen i *Geoinformation, Technology & Management* ved Aalborg Universitet Ballerup i perioden september 2011 til juni 2012.

Specialet tager udgangspunkt i udviklingen af en platformsuafhængig mobil webapplikationen til håndtering af rumlig data. Hovedparten af forløbet er således gået til udvikling af denne. Applikationen kan tilgås online via. Webkit kompatible webbrowsere (fx Google Chrome, Apple Safari og Maxton 3) eller på mobileenheder (fx iPhone, iPad og diverse nyere Andriod enheder) på webaddressen.

www.mapmule.com

(tidsperioden 7.juni – 20. juni hverdage mellem 10-16.)

I specialet er skriftlige kilder angivet efter harvardmetoden, fx (Hansen *et al* 2006). Alle kilder fremgår af litteraturlisten bagerst i specialet, hvor bilagene også vil være at finde. Figurer, kodebider og tabeler i specialet er, med mindre andet er anført, egne.

I forbindelse med udarbejdelsen af projektet vil vi gerne rette en særlig tak til Alper Dinçer, Jason Sanford(GeoJason) og Morten Fuglsang for teknisksparring og input. Endelig vil vi også gerne takke alle på kontoret (lokale 1.1.67) for hjælp til tests, kaffebrygning, hyggelig omgang og stor tålmodighed.

Peter og *Morten*

2. Summary

In recent years, smartphones and tablets have become a trivialized part of every day life. No matter where you are access to the digital world is ensured by the ubiquitous wireless data connection. Network and hardware is continuously getting faster and faster (Danmarks Statestik 2012). The development is furiously pressing on - the future is online!

The digital lifestyle puts a focus on both the opportunities and needs for efficient solutions that take advantage of the opportunities of the digital age. E-Government is a good example of how digital solutions can be used to streamline and optimize workflows. Sharing of data and use of information should be promoted and increasingly used as it will help to break down the silo approach and foster collaboration (Hvingel & Hansen 2011). The interconnection of IT systems and records helped ensure Denmark a place among the world's leading digital nations (Department of Economic and Social Affairs 2012).

There are still plenty of development opportunities within e-Government - especially when it comes to the use of spatial data, which only recently became part of the digital agenda (Regeringen *et al* 2011). The use of spatial data can help take e-Government to the next level, while being a catalyst for value creation for both businesses and individuals.

There is a large potential for the use of smartphones and tablets as tools in e-Government. Applications such as "Giv et praj" have already had success demonstrating the use of mobile devices and user collaboration(DenOffentligeSektor.dk 2011).

The purpose for this project is to identify the scope for how mobile devices can be used as tools for spatial data and e-Government. The upcoming / current registration and apprising of § 3-områder (protected Danish nature types) (MiljøMinisteriet 2010) is used as a case study in the following paper. This paper seeks to develop a "proof-of-concept application" that will be used for this task.

Indhold

Figurer

Tabeller

Kodebider

3. Introduktion

Indenfor de seneste år har smartphones og tablets for alvor gjort deres indtog og er gået hen og blevet hvermandseje. Dagligdagen er blevet digital - lige meget hvor man befinder sig. Adgangen til den digitale verden sikres via den allestedsnærværende trådløse dataforbindelse, der ligesom den digitale hardware, bare bliver hurtigere og hurtigere (Danmarks Statestik 2012). Udviklingen raser derudaf - fremtiden er online!

Den digitale dagligdag sætter fokus på både mulighed og behov for velfungerende løsninger til udnyttelse af de digitale muligheder. Digital forvaltning er et godt eksempel på, hvorledes de digitale muligheder kan bruges til at effektivisere og optimere arbejdsgange. Her har alt fra digital selvbetjening af den enkelte borger, til nedbrydning af de meget omtalte siloer(Hvingel & Hansen 2011) i den offentlige sektor via samkøring af It-systemer og registre, været med til sikre Danmark en plads blandt verdens førende digitale nationer(Department of Economic and Social Affairs 2012).

Der er stadig masser af udviklingsmuligheder indenfor digital forvaltning – især nå det kommer til brugen af geodata, som først på det seneste for alvor er kommet på den digitale dagsorden (Regeringen *et al* 2011). Brugen af stedsbestemt data kan være med til at løfte digital forvaltning til det næste trin, og samtidig være katalysator for værdiskabelse til både virksomheder og privatpersoner.

Potentialet i udnyttelse af smartphones og tablets som værktøjer i den digitale forvaltning, må også siges at være til stede. Applikationer som eksempelvis "Giv et praj", har allerede med stor succes vist mulighederne for brug af de mobile enheder(DenOffentligeSektor.dk 2011).

Baggrunden for dette projekt er, netop at afdække mulighederne for hvorledes mobile enheder kan bruges som værktøjer i forbindelse med geodata og digital forvaltning. Som casestudie benyttes den forestående/igangværende registrering og opdatering af § 3-områder(Miljøministeriet 2010).

I nærværende rapport søges at udvikle en "proof-of-concept-applikation", som vil kunne benyttes til den forestående opgave.

4. Problemformulering og forskningsspørgsmål

På baggrund af ovenstående introduktion er følgende problemformulering udarbejdet:

> *Hvordan kan der ved hjælp af moderne webstandarder og Open Source løsninger udvikles en mobilapplikation til opdatering og registrering af beskyttede naturtyper (§ 3-områder).*

Som uddybning til problemformuleringen opstilles følgende forskningsspørgsmål.

- I hvilken grad kan en mobilapplikation, baseret på web standarder og open source produkter, opfylde de behovene der er til registrering og opdatering af paragra3-områderne?
- Kan rumligt data indrapporteres fra en mobil platform?
- Hvilke faktorer er afgørende for applikationens performance?
- Hvilke muligheder og begrænsninger har den mobileplatform ift. Digital forvaltning?

Førnævnte forskningsspørgsmål søges besvaret dels gennem erfaring gjort under udarbejdelsen af den egentlige applikation og dels gennem relevante litteraturstudier.

I rapporten gennemgås først relevante teoretiske aspekter, som danner grundlag for valg og fravalg i forbindelse med designet af den egentlige applikation. Herefter beskrives implementeringen af selve applikationen.

Projektet er struktureret og udarbejdet efter Agile-metoden(Agile Alliance n.d.), som er yderst brugbar i forbindelse med softwareudvikling. Først udvikles en basisapplikation, med et minimum af den ønskede funktionalitet. Herefter udvikles så mange yderligere funktioner, som den afsatte tidsramme tillader det.

5. Teori

5.1 Digital forvaltning

Digitaltforvaltning (eGovernment) forstås som dét, at IT benyttes til løsning af forvaltningsopgaver. Det værende sig både digitale selvbetjeningsløsninger, som stilles til rådighed for borgere og virksomheder af forskellige offentlige myndigheder via internettet, men også digitalisering af forvaltningsprocesserne hos de offentlige myndigheder. Her er fokus på at effektiviserer interne arbejdsprocesser ved hjælp af IT.

I det følgende kigges der lidt nærmere på digital forvaltning i Danmark, og især på hvilken rolle geodata spille i denne forbindelse

5.1.1 Digital forvaltning af Danmark

I Danmark kom digitalt forvaltning på dagsordenen i midten af 90'erne (Hansen *et al* 2006). Og i starten af det nye årtusinde nedsattes "Den Digitale Taskforce" bestående af repræsentanter fra både statslige og kommunale organisationer. Gruppen udarbejdede den første danske strategi for digital forvaltning *"På vej mod den digitale forvaltning – vision og strategi for den offentlige sektor, januar 2002"*.

I takt med udviklingen indenfor digital forvaltning, er der løbende blevet udarbejdet nye strategier. Det er blevet til fire i alt, hvor den seneste (Den digitale vej til fremtidens vældfærd) udkom i august 2011 og dækker perioden 2011-2015 (Teknologirådet 2005).

Det målrettede arbejde med at gøre forvaltningen digital har båret frugt. Danmark er i dag et af de førende lande indenfor digital forvaltning, se Tabel 1 og Tabel 2). En samlet fjerdeplads på verdensplan – en tredjeplads i Europa - i FN's måling af digital forvaltning (Department of Ecomonic and Social Affairs 2012), viser at Danmark absolut er fremme i skoene, når det kommer til udvikling af løsninger til digital forvaltning. Det er endda værd at bemærke, at Danmark er kravlet helt op i toppen over de seneste to år (Tabel 2).

Rank	Country	E-Government development Index
1	Republic of Korea	0.9283
2	Netherlands	0.9125
3	United Kingdom	0.8960
4	Denmark	0.8889
5	United States	0.8635
6	France	0.8635
7	Sweden	0.8599
8	Norway	0.8593
9	Finland	0.8505
10	Singapore	0.8474

Tabel 1: E-government development index

		E-gov. development index		World e-gov. development ranking	
Rank	Country	2012	2010	2012	2010
1	Netherlands	0.9125	0.8097	2	5
2	United Kingdom	0.8960	0.8147	3	4
3	Denmark	0.8889	0.7872	4	7
4	France	0.8635	0.7510	6	10
5	Sweden	0.8599	0.7474	7	12
6	Norway	0.8593	0.8020	8	6
7	Finland	0.8505	0.6967	9	19
8	Liectenstein	0.8264	0.6694	14	23
9	Switzerland	0.8124	0.7136	15	18
10	Germany	0.8079	0.7309	17	15
	Regional Average	0.7188	0.6227		
	World Average	0.4882	0.4406		

Tabel 2: E-government udvikling fra 2010-2012

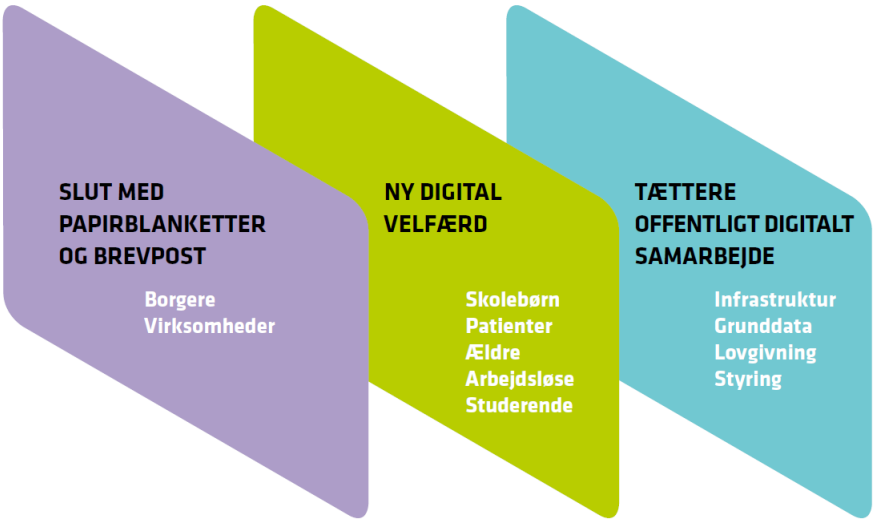
Et af hovedformålene med den danske digitale forvaltning er, at skabe en mere sammenhængende forvaltning. Tanken er at skellet mellem stat, region og kommune og de enkelte forvaltninger skal nedbrydes (Teknologirådet 2005). Dette gøres helt konkret ved, at få de forskellige sagsbehandlingssystemer, til at udveksle data med hinanden. På den måde behøver man ikke lagre alle registre i én stor database, men opnår derimod en netværksstruktur hvor samkøring mellem systemerne sker via fri dataudveksling. Siloerne brydes ned og der skabes én indgang til den offentlige forvaltning (Teknologirådet 2005)

Målet med den digitale forvaltning er dels at levere bedre kvalitet og services til borgere og virksomheder, men også på sigt at hente en økonomisk gevinst på baggrund af medfølgende effektivisering (Teknologirådet 2005). Som eksempel kan tages at samme data tidligere blevet produceret sideløbende flere forskellige steder i de offentlige systemer. Det er slut nu! data skal kun produceres ét sted. Derved opnås både en besparelse i forhold til dataproduktion og vedligeholdelse, men vigtigst af alt vil forvaltningen i fremtiden altid bygge på det samme og mest opdaterede tilgængelige data.

5.1.2 Geodatas rolle i digital forvaltning

Med fokus på digital forvaltning i mere end 10 år, kumme man foreldes til at tro, at tingene bare kørte på skinner. Men der er stadig masser af udviklingsmuligheder i den danske digitale forvaltning.

Eksempelvis kan det vække undren, at det først er i den seneste strategi fra august 2011, *"Den digitale vej til fremtidens vældfærd"* , at grunddata(herunder geodata) for alvor nævnes, som en faktor i forbindelse med digital forvaltning se Figur 1. Dette kan skyldes at der kan være en form for barrierer i forbindelse med anvendelsen af geodata. Hurtig og nem adgang til relevant data, datakvalitet og pålidelighed, samt manglen på målrettede og brugervenlige GIS-værktøjer har været nogle af de helte store bekymringer (Hansen *et al* 2006).



Figur 1: Fokusområder for digital forvaltning (Regeringen *et al* 2011)

Nu er fokus dog endelig kommet til geodataområdet. Gennem både INSPIRE-direktivet og vedtagelsen af *"Lov om Infrastruktur for Geografisk Information"* er der nu for alvor sat fokus på geodata som en hjørnesten i den danske digitale forvaltning (Hansen *et al* 2006)

I den seneste digitaliseringsstrategi fokuseres på at forbedre grunddata om elementer, som eksempelvis matrikler, stednavne, vejnet, vandløb og søer, samt ting som ejendomme, adresser og bygninger (Regeringen *et al* 2011).

Et konkret eksempel på at arbejdet med forbedring af grunddata allerede er i gang, er Miljøministeriets igangværende projekt om opdatering af §-3 områder. Fornyeligt vandt et projekt i forbindelse med dette digitaliseringsprisen 2012 (effektiviseringsprisen). Miljøministeriet havde udviklet en applikation til tablets, som, ved hjælp af GPS og trådløs dataoverførsel, sætter den enkelte medarbejder i stand til hurtigt, pålideligt og effektivt at indberette ændringer eller tilføjelser til det eksisterende datasæt.

Løsningen har resulteret i en forbedret datakvalitet, men også en resursebesparelse på op mod 25%.Ved at fjerne unødige arbejdsprocesser og mindske muligheden for fejl, har man højnet både datasamlingens kvalitet og aktualitet, hvorved borgere, virksomheder og offentlige organisationer opnår en forbedret service(Ministeriet for Videnskab, Teknologi og Udvikling 2003).

Ovenstående projekt, som blev kåret og præsenteret under udarbejdelsen af dette projekt, er et godt eksempel på hvad fremtiden kan bringe indenfor digital forvaltning.

5.2 Data og datastandarder

5.2.1 Diskrete og kontinuerte data

Geografi kan fundamentalt anskues to måder, enten vha. diskrete objekter eller som kontinuerte flader.

Diskrete objekter repræsenter den geografiske verden som veldefinerede objekter i et ellers tomt rum. Menneskeskabte objekter og biologiske organismer passer godt ind i denne model, da det er forholdsvis enkelt, fx at tælle antallet bygninger eller beboere i en by. Her er altså tale om veldefineret data, der let kan sættes i tabeller, og gives attributter. Det er i midlertidigt ikke al data, der er helt så veldefineret. Fx kan det være svært, at definere hvad der er et bjerg, og hvad der er en bakke, eller hvor et bjerget starter og slutter, eller om et bjerg der har to tinder skal defineres som et eller to bjerge.

Geografiske objekter er defineret af deres dimensioner. Polygonobjekter, fx søer og marker, optager et todimensionale areal. Linjeobjekter, fx floder og veje, beskrives som endimensionale linjer. Mens punktobjekter, fx individuelle lygtepæle eller fugle, er nuldimensionale. Diskrete objekter er en meget effektiv måde, til at repræsentere geografisk information om objekter i et todimensionalt rum. Modellen er i midlertidig ikke velegnet til alle typer af data, da den virkelige verden naturligvis er tredimensionel, og da der ikke er steder optaget af intet.

Kontinuerte flader repræsenter den virkelige verden, som et endeligt antal variable, defineret ved enhver given position. Til forskel fra diskrete objekter, hvor en del af det repræsenterede område kan være tomt, er kontinuerte flader altid totalt dækkende. Kontinuerte flader i geografien kan repræsentere mange forskellige ting, fx elevation, overfladetemperatur, **N**ormalized **D**ifference **V**egetation **I**ndex(NDVI) osv.. Kontinuerte flader er defineret af hvad der varierer (fx højde) og hvor jævnt denne varierer.(Longley *et al* 2004)

5.2.2 Metadata og datakvalitet

I det forhåndenværende projekt spiller geografisk data, og behandlingen af denne, en central rolle. Det er derfor vigtigt at have en forståelse, af hvordan geografisk data behandles ift. datakvalitet og metadata.

Metadata er data om data, og beskriver hvordan data er indsamlet og efterfølgende behandlet. Denne information er afgørende, for alle der arbejder professionelt med geografisk data , da metadata fortæller brugeren hvilken nøjagtighed, der kan forventes af data. Uden metadata er det umuligt, at vide om data egnet til en given opgave eller ej. Godt metadata indeholder informationer som fx:

- Hvad repræsenterer datasættet?
- Hvilken indsamlingsstrategi blev benyttet?
- Hvem har indsamlet data? Er der det en annerkendt autoritet?
- Hvorfor blev data indsamlet?
- Hvilken tidsperiode repræsenterer data?
- Hvilket geografisk referencesystem blev brugt ved indsamlingen?
- Hvor god er præcisionen?
- Kan data betragtes som komplet?

Udover at være afgørende for at kunne vurdere, hvor egnet data er til en given opgave, spiller systematisk struktureret metadata også en essentiel rolle ift. rumlige data infrastrukturer. (Cote n.d.)

Fx kan man på den danske geodataportal(http://www.geodata-info.dk) søge efter geodatasæt og geodatatjenester(metadata), opstillet i henhold til gennemførelsesbestemmelserne for INSPIRE-direktivet(forordning nr. 1205/2008 med hensyn til metadata). (Geodata-info n.d.).

5.3 Datahåndtering

Inden for WebGIS findes en lang række dataformater, som hver især har deres egene karakteristika. For bedst muligt at kunne udnytte disse, er det nødvendigt at have en ide om hvad de forskellige formater kan og ikke kan. Her beskrives et par af de mere udbredte formaters fordele og ulemper

5.3.1 WMS

***W**eb **M**ap **S**ervice*(WMS) er en standard for internet kortservices, specificeret af ***O**pen **G**eospatial **C**onsortium*(OGC) i 1999. WMS kræver kun meget lidt af brugeres enhed(PC, tablet osv), og vil derfor kunne fungere, i langt de fleste brugeres almindelige internetbrowsers.(Balstrøm *et al* 2006)

WMS specificerer fem forskellige forespørgselstyper, hvor af to er krav, for at opfylde standarden for WMS servere.

- GetCapabilities – returnerer parameter om WMS'et og de tilgængelige lag
- GetMap – Returnerer et kort i raster format, samt paramenterne for lagene

Udover de obligatoriske kald, kan en WMS server også valgfrit inkludere forespørgslerne

- GetFeatureInfo
- DescribeLayer
- GetLegendGraphic

(Open Geospatial Consortium Inc., 2006)

WMS er et meget udbredt format til håndtering af kort - og rumlig data via internettet, såvel

som til at indlæse data direkte i traditionelle GIS. WMS standarden understøttes bla. af flere *open source* løsninger som fx *MapServer*, *GeoServer* og *GeoZilla,* samt flere kommercielle udbydere, fx *ESRI ArcGIS*, *Google Earth*, *MapInfo Professional* mfl.

5.3.2 WFS.

***W**eb **F**eature **S**ervice*(WFS) specifikationen er en platforms uafhængig standard, defineret af OGC, til transaktion af rumlige features over internettet. I modsætning til WMS der overfører raster filer, kan WFS håndtere vektordata. Data overføres via ***G**eography **M**arkup **L**anguage*(GML), der er en dialekt af det udbredte internet opmarkeringssprog *e**X**tensible **M**arkup **L**anguage*(XML), men andre formater, som fx ESRIs *Shapefiles* og *KML*, er også understøttet.

WFS specifikationen definerer et interface til datamanipulation af geografiske features. deriblandt:

GetCapabilities	Returnere en list over serverens data, og hvilke WFS operationer og parameter der undestøttes
DescribeFeatureType	Returnere information og attributter om et givent datasæt
GetFeature	Returnere den faktiske data, inklusiv geometri og attribut værdier
LockFeature	Forhindre en feature i at blive editeret
Transaction	Editere en eksisterende featuretype ved at oprette nye, opdatere eller slette
GetGMLObject	(kun Version 1.1.0) Returnere elementer ved tværgående Xlinks der referrere til deres XML Ids

Tabel 3: WFS operationer

Basis-WFS understøtter forespørgsler og returnering af features. *Transactional*-WFS (WFS-T) tillader også oprettelse, sletning og opdatering af features. (Geoserver 2012)

WFS kan ses lidt som ”kilde koden” til de raster kort(WMS) der ofte vises online. I stedet for blot at kunne se et billede af et kort, er det med WFS muligt for brugeren selv, at bestemme hvordan data skal visualiseres. WFS tillader ligeledes at Data downloades, analyseres og kombineres med andet data. WFS-T gør yderligere, at WFS er i stand til at understøtte kollaborative kort applikationer. (Geoserver 2012)

WFS er altså et meget dynamisk format, der giver brugeren en masse muligheder. I midlertidigt

er WFS ikke ret godt til at håndtere meget store datamængder, da vektor data og GML kræver en del behandling på klient siden. Ligeledes stiller WFS større krav, til at brugeren har en god forståelse for geodata.(Batty 2010)

5.4 Databaser

Grundlæggende er en database en samling integreret/sammenhængende data om enheder eller helheder. Når der i dag tales om databaser, er der oftest tale om digitale databaser, men databaser har eksisteret længe før computeren, i fx biblioteks kartoteker og telefonbøger.

Rumlig databaser er som udgangspunkt ikke meget forskellige fra traditionelle databaser. Oftest er rumlig database systemer (SDBMS – ***S**patial **D**atabase **M**anagement **S**ystems*) blot traditionelle relationel database system(RDBMS - ***R**elationel **D**atabase **M**anagement **S**ystems*) udvidet, til at kunne håndtere rumlig data. At udvide disse systemer til at kunne håndtere rumlig data, stiller dog en række interessante udfordringer.(Heywood *et al* 2006)

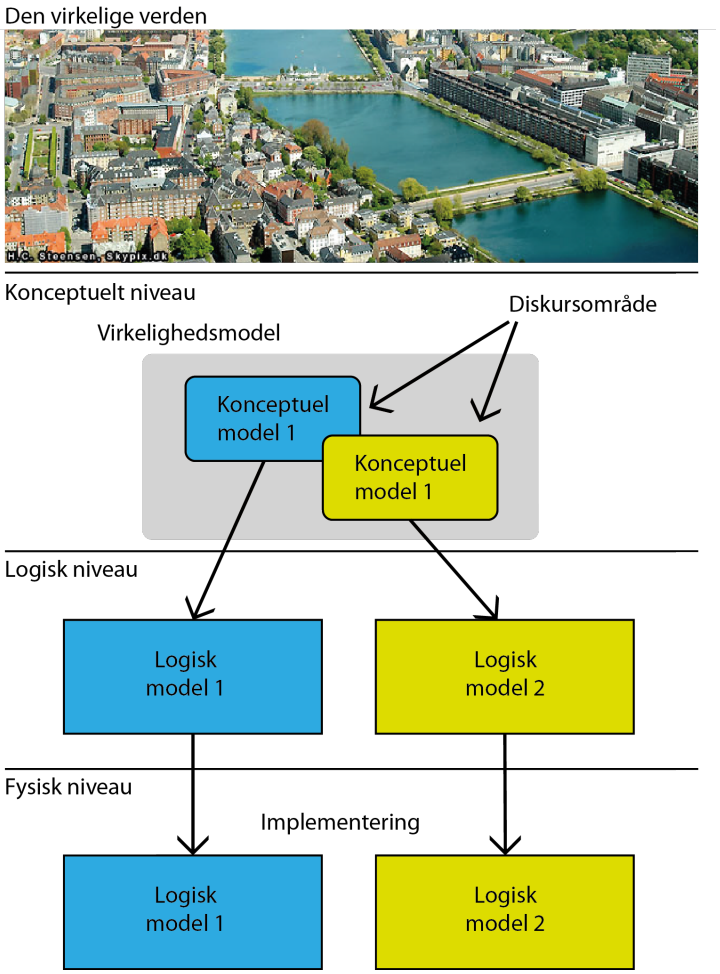
5.4.1 Overordnet databasearkitektur

For at kunne diskutere databaser og databasesystemer på en kort og koncis måde, er det nødvendigt at have en overordnet forståelse af databasearkitektur. Det er værd at fastslå, at der findes flere forskellige teoretiske tilgange, til beskrivelse af modelbegrebet for databasearkitektur. I denne opgave har vi valgt at arbejde med den model, der oftest benyttes i relation til GIS, og som bygger på en ANSI-standard (***A**merican **N**ational **S**tandards **I**nstitute*) (Balstrøm *et al* 2006).

Forståelsen bygger på en firdeling af modelbegrebet, se Figur 2. Den “virkelige verden”, den “konceptuelle model”, den “logiske model” og den “fysiske model”. Den konceptuelle model beskriver hvordan en given del af den virkelige verden kan tolkes, generaliseres og skitseres. Den givne del af virkeligheden kaldes også for diskursområdet(eng. ”universe of discourse”).

Efter den konceptuelle model er fastlagt, kan den logiske modellering begynde. Her arbejdes med forskellige datamodeller, der kendes fra diverse databasesystemer, som fx *rellationelle* – og *objekt orienterede* databasesystemer. Den logiske modellering omfatter også en fastlæggese af datarepræsentation, samt en cirka størrelse på denne.

Endelig er der det fysiske niveau. Dette kommer i forbindelse med fysiske implementeringen af modellen. Her arbejdes der bl.a. med indeksering og den konkrete lagring af data på et digitalt medie. (Balstrøm *et al* 2006).



Figur 2: Illustration af modelbegrebet(Balstrøm *et al* 2006)

5.4.2 Filbaseret lagring

Bag ethvert større GIS findes der en rumlig database. Uden disse databaser ville helt basale GIS funktioner som datalagring og forespørgsler(eng. queries) ganske enkelt ikke være mulige.

I en computer kan lagring af data foregå på to måder. Enten kan data anbringes direkte i filer, hvilket kaldes filbaseret lagring, eller der kan benyttes et såkaldt Database Management System (DBMS).

Benyttes filbaseret lagring, overlader man funktioner og håndtering af data til computerens styresystem (fx Microsoft Windows, Apple OS-X eller Linux). Denne form for filehåndtering har både fordele og ulemper. Så godt som alle moderne styresystemer har i dag en grafisk brugergrænseflade, hvilket gør det let for brugeren, at kigge rundt i filstrukturen, kopiere, flytte og i anden vedligeholdelse af filstrukturen. Indtil 1984 fandtes denne forholdsvist intuitive grafiske tilgang til filhåndtering i midlertidigt ikke, og datamanipulation og –håndtering var en besværlig proces fyldt med fælder. Selv i dag, hvor de grafiske brugergrænsefalder gør det let at skabe sig et hurtigt overblik over filstrukturen, kan der nemt opstå forholdsvis farlige situationer. Eksempelvis hvis man får lyst til at rydde lidt op i computerensfilsystem. Eftersom filerne ligger i en hierarkisk struktur, vil det emner i underliggende mapper forsvinde i det øjeblik en overmappe slettes.(Balstrøm *et al* 2006).

Yderligere er det svært i et filbaseret lagringssystem, at danne referencer mellem forskellig geografisk information, som fx kortet og dets tematiske indhold. Man kan se på det sådan, at sammenhængen kun er tilstede, i det øjeblik alle relevante filer er indlæst i et givent GIS. Det er således individuelle grafiske elementer, der arbejdes med, og en dynamisk sammenhæng mellem to eller flere elementer er ikke mulig.

På trods af de mange uhensigtsmæssigheder ved filbaseret lagring, har netop denne form for datalagring i mange år været de facto standarten for langt de fleste geografiske informationssystemer. I disse år kommer denne tilgang dog under pres fra de mere avancerede og krævende rumlige databasesystemer (SDBMS)(Balstrøm *et al* 2006).

5.4.3 Database Management Systems

Har man prøvet at arbejde med store datamængder, vil man vide at der ikke skal ret mange data eller brugere til, før brugen af filbaseret lagring bliver for besværlig, og risikoen for fejl for stor. Skal mange data og mange brugere væres tilknyttet en database, er der brug for et reelt database management system (DBMS). DBMS er et overordnet begreb, der beskriver en række forskellige databasekoncepter, der i første omgang er defineret af databasernes interne datamodel. Der findes altså en lang række forskellige typer af DBMS, hvor nogle er mere udbredte end andre. Se nærmere beskrivelse af udvalgte DBMS senere i dette afsnit.(Balstrøm *et al* 2006).

DBMS tilgangen til håndtering af geografisk data har en række fordele, i forhold til den traditionelle filbaserede datahåndtering.

- Der kan lettere dannes reference mellem geografisk informationer, som fx kortet og dets tematiske indhold
- Data samles ét sted, hvorved redundans(gentagende data) reduceres
- Vedligeholdelsesomkostningerne nedbringes pga. bedre organisering og mindre dobbelt data
- Applikationer bliver uafhængige af data, hvorved flere applikationer kan benytte det samme data, og udvikles uafhængigt over tid.
- Viden kan lettere overføres mellem applikationer, da databasen forbliver konstant
- Filedeling kan administreres og kontrolleres af såvel administratorer som brugere efter behov
- Sikkerhed og standarter for data og datatilgang kan etableres og håndhæves
- DBMS er bedre i stand til at håndtere mange brugere, der sideløbende tilgår store mængder data

Der findes dog også et par ulemper ved DBMS i forhold til filbaseret datahåndtering.

- Prisen for at anskaffe og vedligeholde et DBMS kan være meget høj
- Et DBMS tilføjer et nyt lag kompleksitet til mindre projekter
- Enkeltbrugerydelsen for filer vil til tider være bedre. Dette gælder særligt for komplekse datatyper og -strukturer.

(Longley *et al* 2004)

Der blev udviklet en lang række forskellige DBMS op gennem 60-90'erne, der alle har forskellige fordele og ulemper. Bl.a. kan nævnes hierarkiske -, netværks -, relationelle -, objektorienterede – og objektrelationelle DBMS. Alle disse vil ikke blive berørt i den forhåndenværende opgave, istedet fokuseres der på de DBMS der har særlig relevans for håndtering af rumlig data. Navnlig relationelle - og objektrelationelle DBMS.

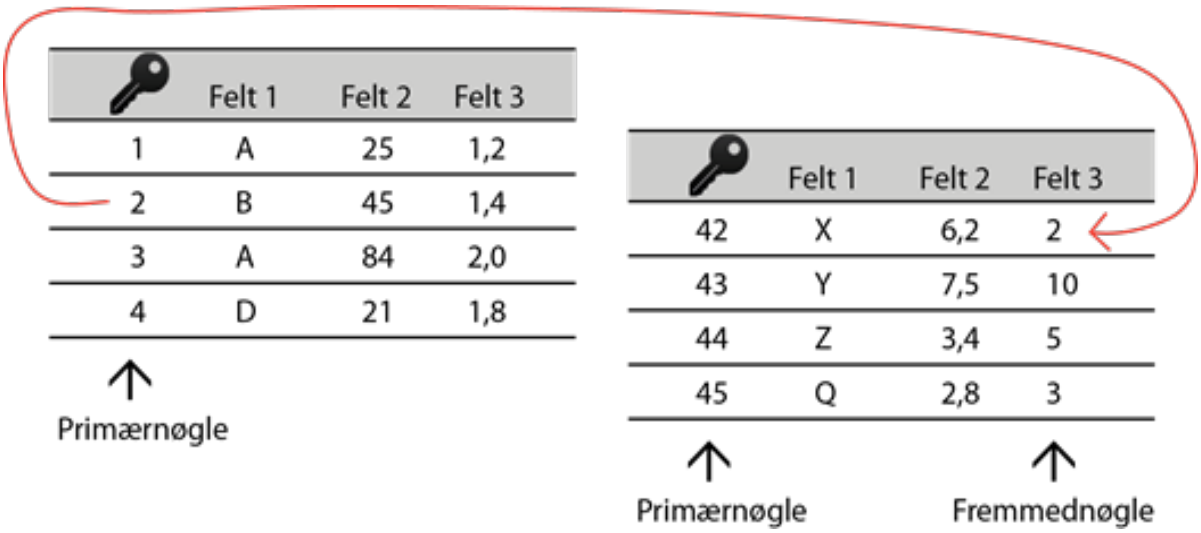
5.4.4 Relationel databasemodel (RDBMS)

***R**elationelle **D**atabase **M**anagement **S**ystems*(RDBMS) er de mest udbredte databasetyper, og ligger bag næsten alle større hjemmesider og computersystemer vi kender. I GIS-regi er denne førsteplads i midlertidigt under pres fra den ***O**bjekt **R**elationelle datamodel* (OR-DBMS), mere om det senere.

I dag er de fleste GIS i stand til, at tilslutte sig direkte til et kommercielt RDBMS. Visse kommer tilmed med deres eget skræddersyede RDBMS. Således benytter GIS ofte RDBMS til at håndtere både attribut - og rumlig data.

Den relationelle datamodel er baseret på et koncept forslået af Edgar F. Codd omkring 1970. Data er organiseret i en række todimensionale tabeller, der hver indeholder information om ét givent emne. Disse tabeller er kædet sammen med fælles data, kaldet nøgler (engelsk: Keys). (Heywood *et al* 2006). Tabel strukturen er meget fleksibel, og tillader at der foretages en langrække forskellige forespørgsler. Ved at linke tabeller sammen med nøgler, er det ligeledes muligt at lave forespørgsler på flere tabeller i et enkelt kald.

Der findes to typer af nøglefelter i en RDBMS, de primære nøgler og de fremmede nøgler. Primære nøgler udgøre en unik identifikation af poster i en tabel. Disse nøgler skal være entydige, og den samme værdi må således ikke forekomme i flere forskellige poster. Fremmede nøgler opstår, hvis man lader en primærnøgle fra en tabel, optræde i en anden tabel, som et ordinært felt, se Figur 9. (Balstrøm *et al* 2006)



Figur 3: Primærenøgler og fremmede nøgler i et RDBMS (Balstrøm *et al* 2006)

Enhver forespørgsel på et RDBMS vil generere nye midlertidige tabeller, på hvilke det også er muligt at foretage forespørgsler. Den relationelle datamodels succes til dels tilskrives systemets store fleksibilitet, men modellen har dog også ulemper. Dels tillader modellen en vis mængde redundans og er lidt langsomere til visse ting, og dels kan den være svær at implementere.(Heywood et al 2006)

5.4.5 Objekt relationel databasemodel (ORDBMS)

***O**bjekt **R**elationel **D**atabase **M**anagement **S**ystem*(ORDBMS) er traditionelle RDBMS, der er tilsat udvidelser, der gør dem i stand til at håndtere komplekse objekter. Begrebet objekt er taget fra den objektorienterede verden, og et ORDBMS kan således ses som en hybrid mellem et RDBMS og et ***O**bjekt**O**rienteret* DBMS (OODBMS).

Fordelene ved at inddrage objekter i et RDBMS er, at de geografiske objekter i databasen foruden at have tilknyttet egenskaber og metadata, også indeholder funktioner til styring af, hvordan objektet skal præsenteres, om specielle forespørgsler og om matematiske funktioner knyttet objektet. (Balstrøm *et al* 2006) Dvs. det er muligt at importere objekter til et GIS, uden der skal foretages yderligere konvertering, transformation eller lignende.

Ifølge Balstrøm *et al* 2006 er en ideal model for et ORDBMS, endvidere i stand til at tage højde for følgende:

- En forespørgselsfortolker (query parser), som udvider den traditionelle fortolkning til også at omfatte de geografiske datatyper
- En funktion til optimering af forespørgsler, så de kan omfatte geografisk relaterede forespørgsler
- Et datamanipulationssprog(DML) som kan håndtere geografiske argumenter. Oftest en udvidet form for ***S**tructured **Q**uery **L**anguage*(SQL)
- En indekseringsmekaniske der kan håndtere ikke kun éndimensionale, men også flerdimensionale geografiske datatyper
- Forbedrede funktioner til lagring af store datamængder. Det er sådan, at geografisk og topologisk data som regel fylder meget(relativt set), og at de derfor kræver mere plads og højere ydeevne af databasen
- Forbedret båndbredde i systemet så transaktioner kan foregå mere smidigt og fordi geografiske data er relativt komplekse
- Forbedrede funktioner til editering af databaser, så objekter, der rettes af brugere, konsekvent bliver rettet i alle dele af systemet

ORDBMS og rumlige databasesystemer er desværre sjældent så komplette i virkeligheden, og der er derfor en del forskellige bud, på hvordan rumlige data håndteres. Der er dog en global udvikling i gang, for at specificere standarter for rumlige databaser, fra både ***I**nternational **S**tandards **O**rganisation*(ISO) og OGC.

5.4.6 Rumlige databasesystemer (SDBMS)

De ovennævnte DBMS har spillet en afgørende rolle i udviklingen af vores moderne informationssamfund, og ligger til grund for så godt som alle aspekter af vores moderne digitale livsstil. Det er i midlertidigt kun inden for de seneste få år, at disse DBMS er blevet i stand til at håndtere geografisk data. Dette skifte har fundamentalt ændret måden, hvorpå vi i GIS sammenhæng kan benytte DBMS.

Traditionelt bliver DBMS primært brugt til at håndtere forretnings- og regnskabsmæssige informationer. Denne type opgaver er RDBMS yderst velegnede til at håndtere, da indekseringen gør systemet i stand til, at kunne håndtere meget store mængder data meget hurtigt. Fx vil et traditionelt RDBMS meget hurtigt kunne give et svar på en forespørgsel(engelsk: query) i stil med ”liste over top 10 kunder, i 2011”.

Giver man i midlertidigt et traditionelt RDBMS en tilsyneladende simpelt rumlig forespørgsel alá ”list alle kunder, der bor inden for 100km af hovedkontoret”, vil et RDBMS’et komme til kort. Grunden er, at de traditionelle RDBMS ikke er i stand til at indeksere rumlige objekter. Skal et RDBMS kunne svare på den rumlige forespørgsel, er det nød til først, at transformere adresserne for hovedkontoret og alle kunderne til et passende referencesystem (fx længde – og breddegrader). Hvorefter den må køre gennem alle emnerne, og beregne afstanden mellem hovedkontoret og hver eneste kunde.

Ønsker man at arbejde med rumligt data, er der altså et behov for en løsning, der er optimeret til at kunne lagre og søge data der er relateret til rumlige objekter. Disse løsninger bliver under et kaldet ***S**patial **D**atabase **M**anagement **S**ystems*(SDBMS)(Shekhar & Chawla 2003).

Som nævnt i forgående afsnit (“5.4.5 Objekt relationel databasemodel (ORDBMS)”) er SDBMS ofte RDBMS, der er optimeret til at kunne håndtere rumligt data som fx punkter, linjer og polygoner. Traditionelle DBMS er kun i stand til at håndtere forskellige numeriske – (engelsk: integer,

floating point mfl.) og tekst datatyper(engelsk: characters, varying characters mfl.), imens SDBMS er i stand til også, at håndtere rumlige datatyper. Disse datatyper er typisk de, der kan findes i *Simple Feature Specification*, udviklet af OGC. Der findes dog flere SDBMS, der også er i stand til at håndtere proprietærer datatyper.

Et SDBMS er ved hjælp af rumligindeksering, og en bred vifte af rumlige SQL-funktioner, i stand til hurtigt at foretage rumlige forespørgsler. Disse rumlige SQL-funktioner er bygget oven på de typiske SQL-kommandoer, såsom fx `SELECT`, `FROM`, `AND`, `OR` mfl. Herunder er de overordnede forespørgselstyper, defineret af OGC (Open Geospatial Consortium Inc. 2006).

- **Spatial Measurements**: Finder afstanden mellem punkter, polygon areal osv.
- **Spatial Functions**: At modificer eksisterende emner til at lave nye. Fx i forbindelse med oprettelse af buffer eller ved udpegning af sammenfaldende emner (engelsk intersecting features)
- **Spatial Predicates**: At foretage booleske(sand/falsk) forespørgelser som fx "findes der boliger inden for 500m af det areal hvor vi planlægger at bygge en motorvej?"
- **Constructor Functions**: At oprette nye emner vha. SQL-forespørgsler, der specificerer nodepunkter (engelsk: vertex) for en linje. Hvis det første og det sidste nodepunkt i SQL-strengen er identiske vil det nye emne kunne være af typen polygon (en lukket linje)
- **Observer Functions**: Funktioner der returnerer specifikke informationer om et emne, som fx placeringen af en cirkels center.

5.4.7 Forespørgsler og SQL

Når der kommunikeres med en database, fx når der foretages søgninger, etableres nye databaser, slettes, redigeres osv. foregår det i alt overskyggende grad vha. SQL. SQL var et af de første kommercielle sprog, brugt i forbindelse med de RDBMS, bygget på relationel algebra og tabelberegning.

I 1986 gjorde ANSI SQL til standard for RDBMS(Seib & Florin 2003), og i 1987 gjorde ISO det samme. Siden har SQL standarden været revideret flere gange, hvilket har udmøntet sig i adskillige forskellige dialekter/versioner indenfor SQL(SQL-86, SQL-89, SQL92, SQL:1999, SQL:2003, SQL:2008 og SQL:2011). De mange dialekter er med til at skabe problemer, når SQL-kode forsøges importeret mellem forskellige DBMS. Det kan derfor være en god ide at undersøge hvilken version af SQL ens DBMS benytter.

SQL kan opdeles i en række delsprog: ***D**ata **D**efinition **L**anguage*(DDL), ***D**ata **M**anupulation **L**anguage*(DML) og ***D**ata **C**ontrol **L**anguage*(DCL).

DDL benyttes til at definere nye objekter. Dette gøres vha. de tre grund kommandoer: CREATE, DROP og ALTER. Med CREATE oprettes nye objekter/tabeller i DBMS'et, se nedenstående eksempel, Kodebid 1.

```
CREATE TABLE kunder{
        Kunde_ID INT NOT NULL,
        Navn CHAR(50),
        Adresse CHAR(50),
        PRIMARY KEY (Kunde_ID)
}
```

Kodebid 1: Oprettelse af tabel vha. DDL SQL

Med DROP sletter et givent objekt fra DBMS og med ALTER laves der grundlæggende ændringer på tabelfelter, datatyper mm. Bemærk at ALTER er ikke en standard funktionalitet.

DML benyttes til at arbejde og manipulere direkte med data. Der er fire grundlæggende kommandoer i DML; INSERT, DELETE, UPDATE og SELECT.

```
INSERT INTO kunder{
(Kunde_ID, Navn, Adresse) VALUES( 123, ”Hans Hansen”, ”Søborg hovedgade 1 Søborg”)
}
```

Kodebid 2: Indsættelse af ny data vha DML SQL

I Kodebid 2, indsættes en ny række i tabellen Kunder, vha INSERT kommandoen. Bemærk at der her manuelt defineres en primære nøgle(kunde_id), til værdi "123". Dette vil man ofte ikke gøre, da der er ved manuel indtastning af primære nøgler, er en øget chance for redundans. I stedet lader man oftest databasen selv skrive den primære nøgle. Således er man sikret den primære nøgle er unikt.

```
DELETE FROM Kunder
WHERE Kunde_ID = 123
```

Kodebid 3: Slet data vha DML SQL

Kodebid 3 viser hvordan hele rækker kan slettes vha. DELETE kommandoen

```
UPDATE Kunder
SET adresse = ”Lautrupvang 2, Ballerup”
WHERE Kunde_ID = 123
```

Kodebid 4: Ændringer i database entry vha DML SQL

Kodebid 4 viser hvordan man vha. UPDATE kommandoen, kan ændre specifikke felter i rækkerne i databasen.

Når der søges i en database benyttes SELECT funktionen. Kodebid 5 viser et eksempel på hvordan en SELECT søgning kan se ud. (Balstrøm *et al* 2006)

```
SELECT Navn
FROM Kunder
WHERE Kunde_ID = 123

Svar : Hans Hansen
```

Kodebid 5: SELECT funktion

Foruden de ovenstående standard SQL-kommandoer, bør også nævnes et par af de rumlige SQL-kommandoer, defineret af den førnævnte OGC standard. Bemærk at ikke alle disse er standarder, men kan være proprietærer kommandoer fra et givent SDBMS system.

Nedenstående Kodebid 6 viser et eksempel på en rumlig SQL-kommando, der returner alle objekter fra databasen, hvor der er sammenfald med geometry-A og geometry-B.

```
SELECT * FROM database WHERE ST_Intersects(
geometry-A,
geometry-B
);
```

Kodebid 6: Rumlig SQL forspørgsel

Nedenstående Kodebid 7 viser et eksempel på en rumlige SQL-kommando, der returner alle objekter fra Geometry-A i databasen, inden for radius-X afstand af Geometry-B

```
SELECT * FROM database WHERE ST_DWithin(
       Geometry-A; Geometry-B, radius-X
);
```

Kodebid 7: Rumlig SQL forspørgsel om sammenfaldende punkter

Endelig har vi et eksempel på hvorledes rumlig SQL, kan benyttes til at tilføje rumlige objekter til en database, se Kodebid 8.

```
INSERT INTO kunder
(Kunde_id, Navn, Adresse, geocol)
VALUES( 123, "Hans Hansen", Søborg Hovedgade 1 Søborg, ST_GeomFromText('POINT(lng,
lat)', SRID));
```

Kodebid 8: SQL kommando til at tilføjde data

5.4.8 Indeksering

Jo mere data en database indeholder, des vigtigere er det at have en effektiv indeksering. Et indeks er en måde at organisere data på, der gør det hurtigt og effektivt at finde de ønskede informationer/data. I bøger benyttes et indeks bag i bogen, der oplister ord og begreber fra teksten, hvilket gør det betydeligt nemmere og hurtigere, at finde det man søger. Havde vi ikke disse indeks, ville fx en telefonbog være så godt som ubrugelig, da vi i stedet for et hurtigt opslag, ville være nød til at læse hele telefonbogen igennem, for at finde det ønskede nummer.

I dag er flere og flere indeks flyttet over i computerne, og at finde de bedste indekseringsformer, er blevet en hel videnskab(Balstrøm *et al* 2006). Det anslås at Google i 2011 håndterede ca. 4.717.000.000 søgninger om dagen, eller ca. 54.594 søgninger i sekundet. (Statistic Brain 2012) For at kunne håndtere sådanne astronomiske tal, er store internetsøgemaskiner som Google, Yahoo og Bing, nød til at gøre brug af store maskinparker, der i døgndrift indsamler og indekserer information på internettet via komplekse algoritmer. Google dominans på søgemarkedet, skyldes i høj grad den komplekse algoritme, der ligger til grund for søgemaskinens imponerende præcise og hurtige svar. Svar der bygger på det indeks algoritmen laver.

Indeksering er af særlig afgørende betydning for hastigheden af et SDBMS, da der ofte arbejdes med store datamænger. PostgreSQL m. PostGIS benytter en form for indeksering kaldet ***G**eneralised **i**ndex **S**earch **T**ree* (GiST) til indeksering. En specialiseret indekseringsform, i stand til at indeksere såvel standardobjekter, som rumlige objekter.

5.5 Korttjenester ift. Open Source, API og SDK

Der findes i dag adskillige online korttjenester, fx Google Maps, Microsoft Bing Maps, OpenStreetMap mfl., der alle giver udviklere en eller anden form for adgang, til at benytte deres system i egne produkter, hjemmeside, projekter osv.

Den mest almindelige adgang til korttjenester er et ***A**pplication **P**rogramming **I**nterface* (API). Som navnet antyder er et API en brugerflade, og kan sammenlignes med fx telefonsystemet eller de elektriske installationer i et hus. Alle kan principielt benytte systemerne, så længe brugerfladen er kendt. Du kan fx købe produkter, der gør brug af en API, på samme måde som du kan købe en lampe eller en telefon, der kan tilsluttes hhv de elektriske installationer eller telefonnettet. Et API giver kun adgang til den del af systemet, som udviklerne af det pågældende systemet har valgt. Det er altså ikke muligt for en tredjepart at ændre på selv opbygningen af systemet, men kun at foretage kald til systemet, fx "GetMap".

Visse produkter har en tilknyttet ***S**oftware **D**evelopment **K**it* (SDK). Et SDK er et implimenterings værktøj. Det er kan sammenlignes med et samlesæt, der gør brugeren i stand til at bygge et skræddersyet produkt, der kan tilslutte sig det givne produkt, som fx en korttjeneste eller et smartphonestyresystem.

Hvis et API er veldokumenteret, bør der teknisk set ikke være behov for et SDK, for at kunne udvikle software til brug med API'et. Men at have et SDK gør det generelt noget lettere og mere fleksibelt (Sachs 2012).

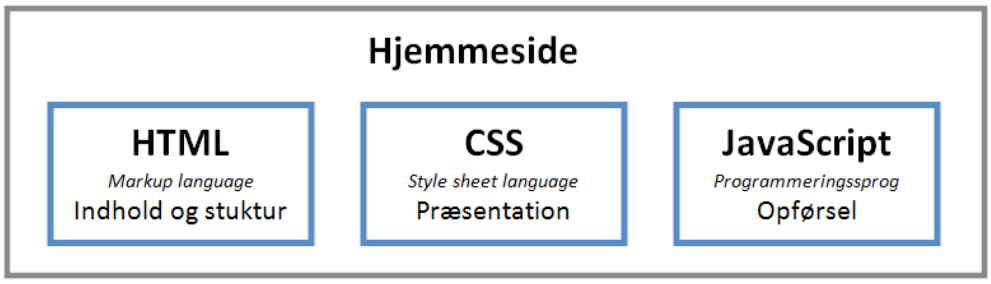
Open Source er et begreb, der dækker over en række softwarelicenser, der har det tilfælles, at de giver brugeren lov til at ændre i programmets kildekode, lave nye versioner af programmet og at videregive disse nye versioner. Det er ligeledes tilladt at lave nye kommercielle version af et Open Source produkt, eller at benytte dele af et Open Source produkt kommercielt. I modsætning til API'er og SDK'er giver Open Source altså brugeren fuld kontrol og ejerskab over sit produkt. Dette er en fundamental forskel og måske den største styrke ved at programere op imod en Open Source løsning.

Programmerer man op i mod et kommercielt produkt, må man være parat på at ejeren, af det pågældende produkt, kan ændre vilkårene fra dag til dag. Dette skete bl.a. med Google Maps, der d. 1 januar 2012 indførte brugerbetaling for storkunder(brugere af google maps api med over 25.000 bru-

gere pr dag) (BBC 2011). Er man ikke klar over at sådanne ændringer kan forekommer, kan man som udvikler ende i en meget uheldig situation, hvor er produkt der tidligere var gratis, pludselig koster penge, eller måske slet ikke er tilgængeligt længere.

5.6 Programmeringssprog

Det er forholdsvis normalt at se ***H**yper**T**ext **M**arkup **L**anguage* (HTML), ***C**ascading **S**tyle **S**heets* (CSS) og ***J**ava**S**cript*(JS), beskrevet, som de tre grundsprog for internetsider, se Figur 4 (Allardice 2011).



Figur 4: De tre grundsprog for internetsider (Allerdice 2011)

5.6.1 Html

HTML er et *Mark-up Language* (opmarkeringssprog), baseret på *Standard Generalized Mark-up Language*(SGML), der er et Mark-up Language til dokumenter, som definerer hvordan tekst skal præsenteres på forskellige fysiske enheder. HTML blev udviklet af to CERN forskere i Schweiz, med Tim Berners-Lee som hovedarkitekt. Formålet med sproget var, at give forskerne et simpelt Mark-up Language, til at præsentere deres videnskabelige resultater på internettet.

Den første beskrivelse af HTML standarden blev udgivet i slutningen af 1991, men den første komplette specifikation blev ikke udgivet før november 1995, samtidig med udgivelsen af HTML 2. Der har efterfølgende flere versioner af HTML, og den officielle HTML standard er lige nu er HTML 4, Iflg. ***W**orld **W**ide **W**eb **C**onsortium*(W3C).

HTML 5 er den næste revision af HTML standarden, og er pt. en *Working Draft* hos W3C. HTML5 tilføjer drastiske ændringer til HTML sproget. Særligt flere nye syntaksfeatures, som fx <video> -, <audio> - og <canvas> elementer, såvel som integrationen af skalerbar vektorgrafik. De nye features er designet til at gøre det lettere at indkludere og håndtere multimedia og grafisk indhold på nettet, uden at være afhængig af proprietære plugins og API'er. I de seneste år har store firmaer, som Apple og Google, gjort meget på for at få HTML5 standarden udbredt. Derfor har HTML5 allerede nu en stor udbredelse, og er implementeret i flere browsere, end man normalt vil kunne forvente af en forholdsvis ny standard.(Kesteren & Written 2012)

HTML bliver i dag brugt på stort set alle internetsider. Sproget gør det muligt at indlejre billeder, objekter og scriptingsprog (fx JavaScript) på sider. HTML kan ligeledes bruges, til at definere en internetsides udseende, men oftest overlades dette til CSS.

HTML dokumenter er opbygget af elementer. Det mest basale HTML dokument består af tre elementer. De første elementer er `<html>` og `</html>` mellem disse er diverse elementattributter samt tekst eller grafisk indhold placeret. Et basalt HTML dokument kunne se sådan ud, Kodebid 9:

```
<html>
    <body>
        <h1>”Hello World”</h1>
    </body>
</html>
```

Kodebid 9: Basis HTML syntakt

Her ses start tagsene `<html>` og `</html>`, element attribut tagsene `<body>` og `</body>` samt tekst indholdet `”Hello World!”`. Bemærk at `”Hello World!”` er mellem tagene `<h1>` og `</h1>`, hvilket fortæller browseren, at der er tale om tekst i første niveau overskrift (World Wide Web Consortium 1, n.d.).

5.6.2 CSS og Sass

CSS er et *Style Sheet sprog*, der benyttes til at beskrive udsenende og formatering(style) af dokumenter skrevet i diverse *Mark-up Languages*(HTML, XML, SVG mfl.). Selv om sproget principielt kan formatere alle XML-baserede sprog men benyttes CSS hovedsageligt til HTML.

CSS kan enten indlejres direkte i et HTML dokument mellem to `<style>`-tags, eller placeres i filer for sig selv. Fordelen ved at placere CSS'en i en fil for sig selv er, at formateringen derved kan genbruges flere steder i samme dokument eller i flere forskellige dokumenter. Placeres CSS'en ikke i en fil for sig selv, vil man være nødsaget til at gentage den samme kode på hver side, hvor man ønsker den samme effekt. En praksis der både tager meget tid, og skaber unødig redundans i koden.(World Wide Web Consortium 4, n.d.)

I et HTML dokument kan CSS implementeres på tre måder. Der kan refereres til en eller flere CSS filer, ved at linke til dem i HTML'et <head>, se Kodebid 10:

```
<link>rel="stylesheet" href="default.css type"test/css">
<link>rel="stylesheet" href="advancedStyle.css type"test/css">
```

Kodebid 10: CSS indlejret som links i HTML <head>

CSS'en kan skrives ind i HTML koden i et <style>-element, se Kodebid 11.

```
<style type="text/css">
 p {
   color: red;
   margin-left: 15px;
 }
</style>
```

Kodebid 11: CSS indlejret som <style> element i HTML'en

Og endelig er det muligt at skrive CSS ind direkte i et HTML elemtent:

```
<p style="color: red; margin-left: 15px">
Hello World!
</p>
```

Kodebid 12: CSS indlejret direkte i et HTML <p> element

Et eksempel på hvordan CSS syntaken ser ud kunne være således, se Kodebid 13.

```
body
   {
   background-color:#d0e4fe;
   }

h1
   {
   color:orange;
   text-align:center;
   }

p
   {
   font-family:"Times New Roman";
   font-size:20px;
   }

h1,h2,h3
   {
   color: #000fff;
   text-align: center;
   }
```

Kodebid 13: Eksempel på CSS syntaks

CSS3 er W3Cs seneste revision af CSS standarden, og er pt. en *Working Draft*. Flere af de fordele der typisk associerer med HTML5, er faktisk forbundet til CSS3 og JavaScript (Robinson 2010). På trods af at HTML5 er en stor revision af HTML standarden, håndter HTML stadigvæk kun indholdsstrukturen. Således er det stadigvæk JavaScript der håndter opførsel og CSS der klare style.

CSS3 bygger på den vidt udbredte CSS2.1 standard, og gør det muligt at lave visuelt imponerende interfaces. En af de mest interessante features i CSS3 er *CSS Animation,* der sammen med CSS Transform, kan understøtte en robust visuel oplevelse på linje med Flash eller selv Native Apps, se også afsnit 5.8 *"Lokale, Web- og hybride applikationer" on page 32*.

***S**yntactically **A**wesome **S**tyle**S**heets*(SASS) er en pre-processor til CSS, der tilføjer nye syntakser til sproget. Heriblandt variable, mixins, nesting og math/color funktioner. For eksempel kan vi i SASS skrive, se Kodebid 14:

```
$blue: #3bbfce;
$margin: 20px;

.content-navigation {
      border-color: $blue;
      color: darken($blue, 9%);
}
.border {
    padding: $margin / 2;
    margin: $margin / 2;
    border-color: $blue;
```

Kodebid 14: SASS syntaks

Hvilket kan complies til følgende CSS kode, se Kodebid 15.

```
.content-navigation {
    border-color: #3bbfce;
    color: #2b9eab;
}

.border {
    Padding: 10px;
    Margin: 10px;
    Border-color: #3bbfce;
}
```

Kodebid 15: CSS compilet fra SASS

Som det ses af overnstående, har SASS ændret variablen `$blue` fra hex farven `#3bbfce`, til den 9% mørkere farve `#2b9eab`. Sass har yderligere halveret margen fra `20px` til `10px` vha. "`$margin / 2`".

Compass udvider SASS yderligere, ved at tilføje forskellige mixins og gøre et udvidelsessystem tilgængeligt. Vha. Compass kan man lave regler som nedentående, se Kodebid 16:

```
$boxheight : 10em;

.mybox {
    @include border-radius ($boxheight / 4);
}
```

Kodebid 16: Compass syntakt

Hvilket compiler Compass koden til følgende CSS, se Kodebid 17:

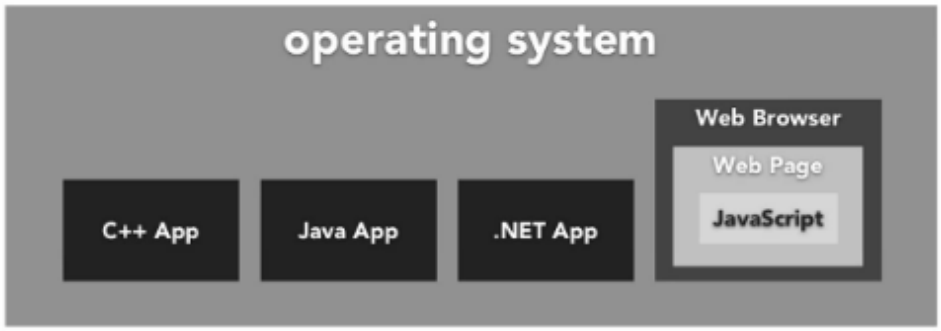
```
.mybox{
    -webkit-border-radius: 2.5em;
    -moz-border-radius: 2.5em;
    -o-border-radius: 2.5em;
    -ms-border-radius: 2.5em;
    -khtml-border-radius: 2.5em;
    border-radius: 2.5em;
}
```

Kodebid 17: Compass til CSS

Bemærk at Compass compiler ".mybox", til at være kompatibel på tværs af mange forskellige browsere, ved at benytte flere forskellige syntakser for "border-radius". (Kaneda 2010)

5.6.3 JavaScript / ECMAScript

JavaScript er et programmeringssprog, men bliver ofte refereret til som et scripting sprog(Allerdice 2011), pga. de begrænsninger sproget har. JavaScript har ikke har de samme egenskaber, som et traditionelt programmeringssprog. Det er fx ikke muligt, at skrive et program direkte til et computerstyresystem i JavaScript, på samme måde som ville med fx *C++*, *Java* eller *.NET*. Det skyldes at JavaScript kun kan fungerer inden i et andet program, navnligt en webbrowseren (fx Internet Explore, FireFox, Chrome osv.). Styresystemet indeholder webbrowseren, webbrowseren indeholder en side, og siden indeholder JavaScript, se Figur 5.



Figur 5: JavaScripts placering i et computersystem (Allardice 2011)

En anden begrænsning ved JavaScript er, at det ikke har adgang til filesystemet på den maskine det kører på. Dette er en bevidst begrænsning, da afgang til filesystemet, ville udgøre en sikkerhedsrisiko. Af samme grund kan JavaScript heller ikke skrive til en database eller tilgå hardware (USB sticks osv.).

JavaScript er et client-side programmeringssprog, og bliver på samme måde som HTML og CSS, sendt til brugerens computer som ren tekst, hvorefter webbrowseren fortolker koden. Dette er modsat serverside programmeringssprog (fx PHP, ASP.NET, Ruby on Rails mfl.), der udfører koden på serveren, og kun sender resultatet tilbage (Allerdice 2011).

JavaScript implementeres på to måder i et HTML dokumententen direkte i HTML koden, se Kodebid 18.

```
<script type="text/javascript">
    document.write("Hello World!");
</script>
```

Kodebid 18: JavaScript skrevet ind direkte i HTML

Eller som et link i HTML'ens `<head>`, se Kodebid 19

```
<head>
<script type="text/javascript" src="…/script.js"></script>
</head>
```

Kodebid 19: Link til JavaScript i HTML <head>

JavaScript syntaksen kunne se ud som følgende, se Kodebid 20.

```
<script type="text/javascript">
var navn = "Jacob";           //Definerer variablen 'navn' med værdien 'Jacob'
document.write(navn);         //Udskriver teksten 'Jacob', som defineret i variablen
navn

navn = "Erik"                 //Erstatter værdien 'Jacob' fra variablen 'navn'
document.write(navn)   ;      // Udskriver teksten 'Erik'

document.write("hello " + navn);      //Udskriver teksten 'hello Erik'
</script>
```

Kodebid 20: JavaScript syntaks

JavaScript biblioteker er biblioteker over allerede skrevne JavaScripts, som udviklere kan kalde frem og genbruge i JavaScript baserede applikationer. Disse gør det lettere for udviklerne, at lave JS applikationer, da alt ikke skal skrives fra bunden. Der findes et hav af forskellige biblioteker, til forskellige formål. Blandt de mest udbredte kan nævnes JQuery, Prototype, Ext Core, isPHP og MooTools.

Til udvikling af WebApps findes der ligeledes specifikke JS biblioteker. Fx Jquery Mobil og Sencha Touch fra folkene bag hhv. Jquery og Ext Core.

5.6.4 PHP

PHP: *Hypertext Preprocessor,* eller blot PHP, er et Open Source scripting sprog, som oprindeligt var designet til at generere dynamisk indhold til webapplikationer. I dag bruges PHP primært til server-side scripting, hvor det bruges til at føde en klient med dynamisk indhold fra en web-server. PHP kan kommunikere med de fleste databasesystemer, ligesom det kan bruges på de fleste platforme, styrersystemer og web-serverer. PHP kan indlejres direkte i HTML-dokumenter og anses desuden for at være et velydende programmeringssprog. PHP har hundredvis af basisfunktioner og derudover findes en lang række tilføjelser/udvidelser.

PHP parser kun kode indenfor dets afgrænsninger. <?php bruges som åbning, mens ?> lukker afgrænsningen. Al kode udenfor afgrænsningerne bliver sent direkte til klienten som output, se Kodebid 21.

```
<?php
    $navn = "Jens Hansen";      // Variablen "navn" tildeles værdien "Jens Hansen"
    echo "Hej ".$navn;          //  Funktion som skriver til skærmen

?>
```

Output: Hej Jens Hansen

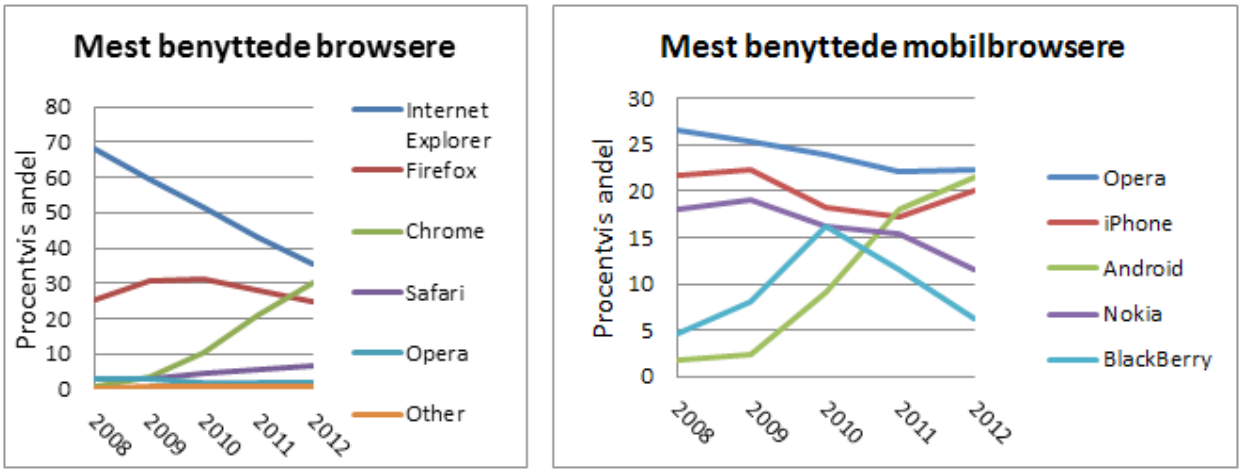
Kodebid 21: PHP syntaks

5.7 Webbrowsere

Når der snakkes web-udvikling, er det vigtigt også at få nævnt web-browsere. En web-browser er en applikation, hvis vigtigste opgave er at præsentere den kode, som programmøren har udarbejdet. Programmørens vigtigste opgave er derfor, at udvikle kode som vises korrekt til brugeren. En opgave som, især tidligere, har været noget af en udfordring, da forskellige browsere har fortolket og vist den samme kode forskelligt.

Heldigvis har W3C udarbejdet standarder for programmering og udvikling af web-sites og HTML. Et initiativ som de seneste år har gjort livet noget lettere for web-udviklerne, efterhånden som disse standarder er blevet integreret i de forskellige browsere.

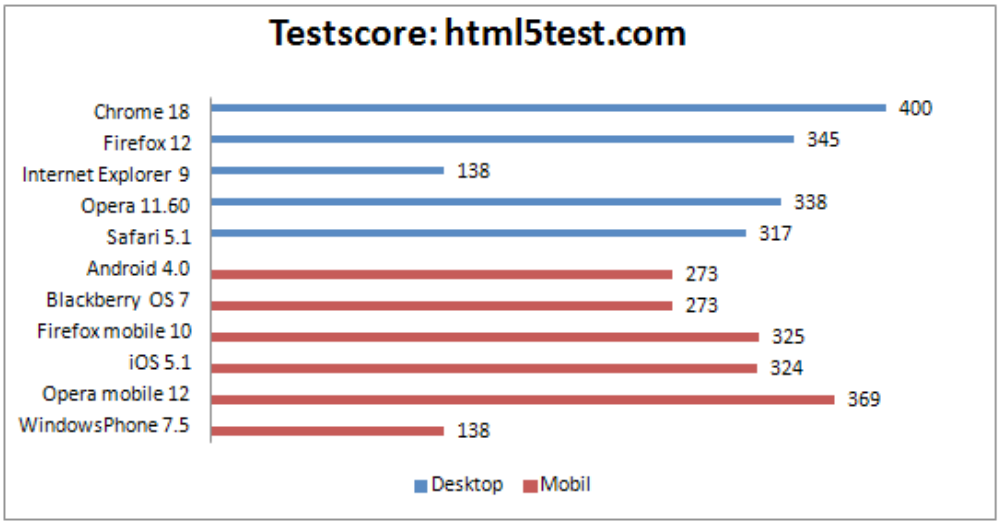
Herunder ses en oversigt over de mest benyttede desktop- og mobile web-browsere, Figur 6.



Figur 6: Mest benyttede browsere 2008-008(tv) og mest benyttede mobilebrowsere 2008-2012(th) (STATCOUNTER, 2012)

Selvom udviklingen med implementering af standarder går den rigtige vej, er der stadig et stykke vej endnu – også for de helt store spillere på markedet. Især når det kommer til nye teknologier/standarder som HTML5 og CSS3.

Nedenstående Figur 7 viser forskellen på hvor godt de nuværende web-browsere er gearet til at tackle de nyeste web teknologier.



Figur 7: Viser forskellen på hvor godt de nuværende web-browsere er gearet til at tackle den seneste HTML5 og CSS3 teknologi (Sights 2012)

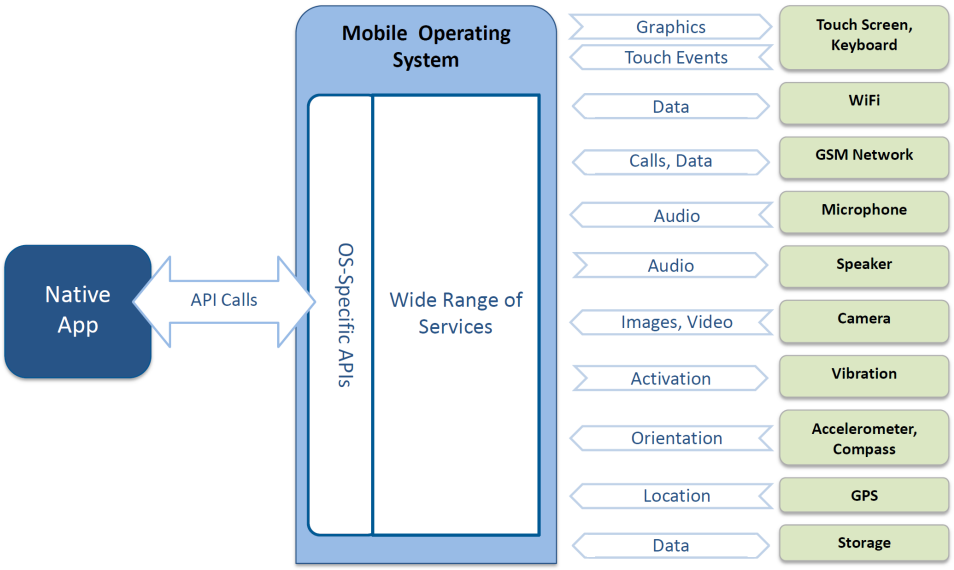
Den grafiske præsentation af web-udviklerens kode styres af browserens layout-motor. Og der er her en grundlæggende forskel på browsere. Internet Explorer benytter nemlig sin egen *Trident* layout-motor, mens *Firefox Gecko*, *Oprera Presto, Chrome* og *Safari* benytter *WebKit* layout standarden. Det er derfor intet under, at der er forskel på hvordan web-sider vises i Internet Explorer ift. de andre store browsere.

Heldigvis går udviklingen stille og roligt mod at flere og flere browsere benytter sig af Webkit løsningen som layout-motor (hvilket allerede i høj grad er tilfældet, når man ser på de mobile browsere). Webkit løsningen er oprindeligt udviklet af Apple, og blev i 2005 gjort tilgængeligt som Open Source.

5.8 Lokale, Web- og hybride applikationer

Udvikling af applikationer til smartphones kan gøres på flere forskellige måder. I det følgende afsnit søges de forskellige muligheder, samt deres styrker og svagheder, belyst

5.8.1 Lokale applikationer



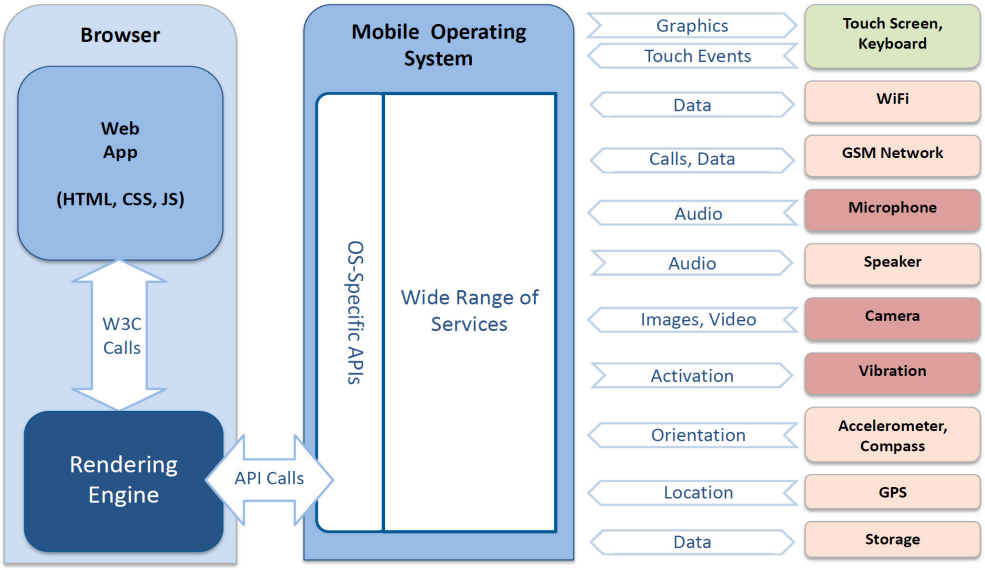
Figur 8: Lokal applikations diagram (WORKLIGHT n.d.)

Lokale applikationer, se Figur 8, er binære eksekverbare billeder som afvikles direkte af enhedens styresystem. Applikationen lagres i enhedens hukommelse og interagerer med enheden ved hjælp af styresystemets API. Dette betyder, at applikationen skal udvikles specielt til det ønskede styresystem og medføre derfor høje omkostninger til udvikling og vedligeholdelse (eksempelvis skal applikationer til Apples produkter programmeres i *Objective-C* mens der til Andriod skal programmeres i Java) (WORKLIGHT® n.d.).

Den klare fordel ved lokale applikationer er, at de har direkte adgang til alle enhedens funktioner (GPS, kalender, kamera etc.) og kan bruges offline (hvis applikationen ellers ikke skal hente data fra eksterne kilder). Dét at applikationen ligger lokalt, sikre en hurtig, flydende og stabil afvikling af applikationen (HIRD 2011).

Applikationerne distribueres via App stores, hvilket sikre synlighed og nem distribution til brugeren. Distribuering via App stores har dog den ulempe, at applikationen skal godkendes af en tredje part (App store ejeren), inden den bliver frigivet i App storen (HIRD 2011).

5.8.2 Web-applikationer



Figur 9: Web app (Worklight n.d.)

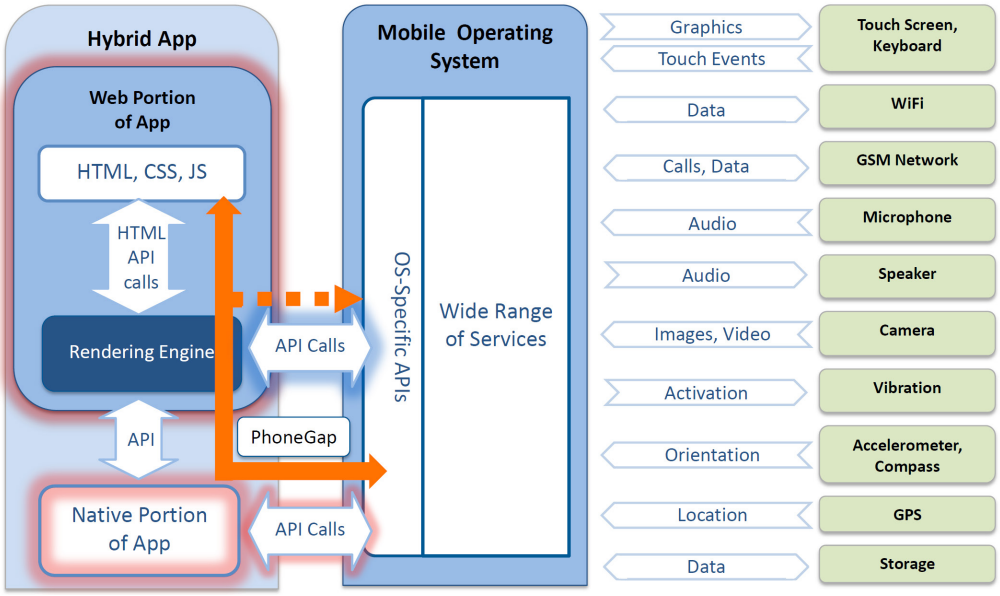
Web-applikationer, se Figur 9, udvikles ved hjælp af web-teknologier (HTML, CSS og JavaScript m.fl.). Applikationen afvikles gennem browseren, og ikke direkte af enhedens styresystem som ved den lokale applikation. Udviklingen ved hjælp af web-teknologier sikrer kompatibilitet med mange forskellige enheder, og er derfor ikke styresystemsafhængig. Da applikationen tilgås via browseren må både selve applikationen og evt. yderligere data hentes via netværket (Worklight n.d.). Det kan resultere i længere opstartstider sammenligned med en lokal applikation.

Web-applikationer har ikke adgang til alle enhedens funktioner. Der bliver dog løbende flere og flere funktioner tilgængelige efterhånden som standarder for adgang til disse implementeres i browserne (via W3C's Mobile Web Initiative)(World Wide Web Consortium 2, n.d.). Web-applikationer optimeres til brug på smartphones og kan derfor under brug have fuldstændig samme udseende som en lokal applikation(Worklight n.d.).

En klar fordel ved web-applikationer er udviklingsomkostningerne og tilgængeligheden. Applikationen kan udvikles af almindelige web-udviklere, hvilket betyder en både billigere og hurtigere udviklingsproces.

Applikationerne distribueres via internettet og der er derfor ikke krav til tredje-parts-godkendelse(som fx i Apples AppStore), inden applikationen gøres tilgængelig for brugeren. Det er heller ikke påkrævet at brugeren skal installere applikationen. Dette gør også elementer som vedlige-holdelse og opdatering betydeligt nemmere. Distribueringen via internettet øger det potentielle antal brugere af den udviklede applikationen (HIRD 2011).

5.8.3 Hybrid-applikationer



Figur 10: Hybrid App (WORKLIGHT n.d.)

Hydrid-applikationen, se Figur 10, er en lokal applikation med indlejret HTML. Applikationen har alle fordelene fra den lokale applikation mht. adgang til enhedens funktioner. Gennem den indle-jrede HTML udnyttes web-applikationens forcer i forhold til kompatibilitet og lavere udviklingsom-kostninger i web-udvikling.

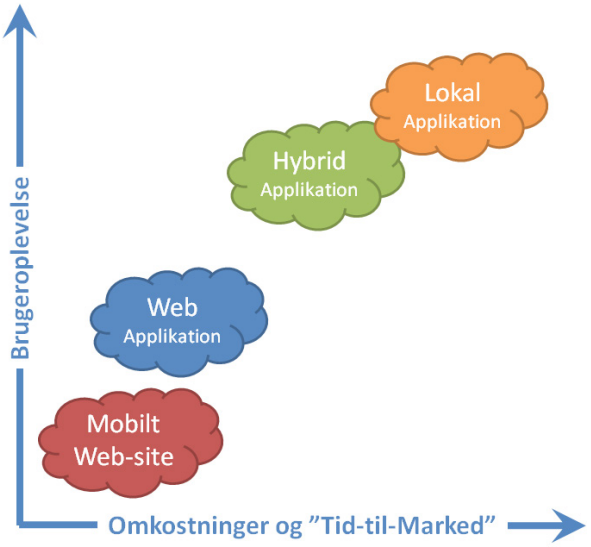
Hybrid-applikationer udvikles ofte ved hjælp af udviklingsværktøjer som eksempelvis Sencha, PhoneGap og Titanium (CAVAZZA 2011). Med disse værktøjer kan applikationen udvikles ved hjælp af web-teknologier hvorefter udviklingsværktøjet omdanner koden til brug i hydrid-applikationer. Der-ved kan udviklingsomkostninger og -tid holdes lavere end ved udvikling af lokale applikationer.

Web-delen af applikationen kan enten tilgås online via et netværk, eller pakkes og lagres med selve applikationen på enheden. Applikationen kan, ligesom den lokale applikation, distribueres via

divs. app stores med de fordele og ulemper det som tidligere nævnt indebærer. Den store forskel ligger i måden udvikleren vælger at håndterer applikationens HTML-indhold på. Vælgers det at HTML-indholdet pakkes og distribueres sammen med applikationen, vil det skulle godkendes af app store ejeren, mens HTML-indhold tilgået online ikke vil skulle godkendes. Pakket HTML-indhold vil betyde bedre performance, mens online HTML-indhold vil være lettere at opdaterer.

5.8.4 Valg af applikationstype

At vælge om en applikation skal udvikles som en lokal-, web- eller hybrid-applikation kan være ganske komplekst. Nedenfor ses et diagram (Figur 11) som illustrerer sammenhæng mellem omkostninger og tid til udvikling i forhold til brugeroplevelsen.



Figur 11: Sammenhæng mellem omkostninger og tid til udvikling i forhold til brugeroplevelsen (Worklight n.d.)

Der er desuden en række ting der bør klarlægges inden der træffes valg om applikationsform:

- Hvilke platforme skal app'en køre på?
- Hvordan skal brugeren nås?
- Hvad er der af behov for opdatering af applikationen?
- Krav til brug af enhedens funktioner
- Hvad er tidshorisonten?
- Hvordan ser økonomien i projektet ud?

Nedenfor ses en grafisk oversigt over nogle af de fordele og ulemper, der kan være ved valg af de forskellige applikationsformer, se Figur 12.

Figur 12: Fordele og ulemper ved valg af app type (Worklight n.d.)

5.9 Lokalisering

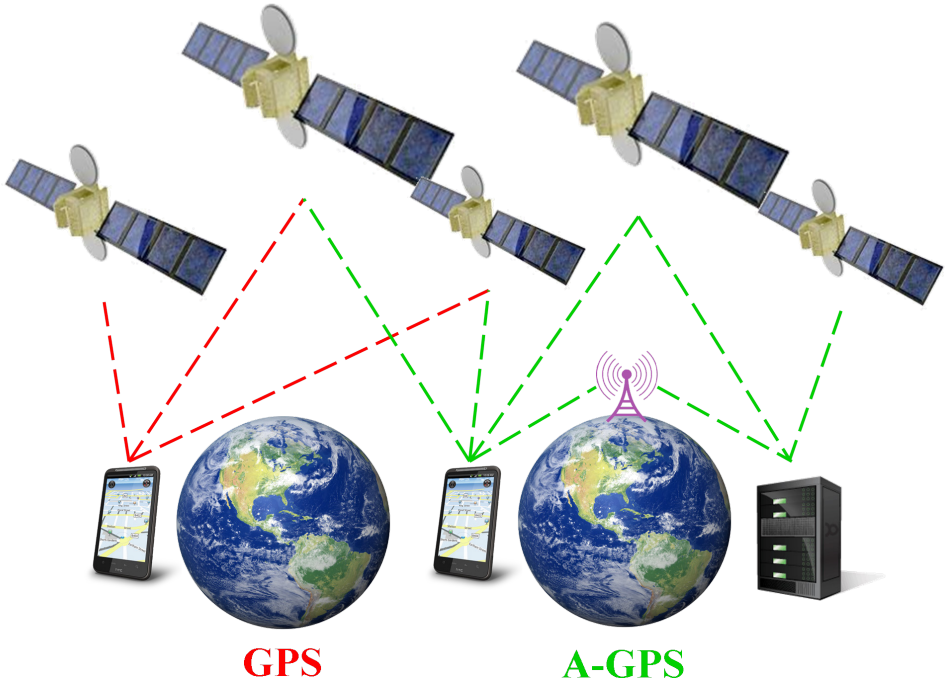
Muligheden for at knytte information til en geografisk placering er nøgleelementet, når der snakkes geodata og digital forvaltning. Der findes en række teknologier, som kan bruges til lokalisering. I det efterfølgende vil være en kort gennemgang af forskellige metoder til at gøre dette. Metoderne beskrives alle med baggrund i et ønske om at få koordinater til en konkret lokation.

Geokodning: Er en simpel og hurtig måde at få adgang til koordinater for en given adresse. En adresse søges i en database indeholdende information om adresser og tilhørende koordinater, hvorefter de ønskede koordinater returneres.

Triangulering via sendemaster: En mobilenhed(med en dataplan) har altid kontakt til en eller flere sendemaster i nærheden. Med kendskab til masternes placering og måling af signalstyrken fra de forskellige master, er det derfor muligt at beregne enhedens fysiske placering. Beregningen klares hurtigt, mens nøjagtigheden, som afhænger af antallet af master og afstanden til dem, må forventes ikke at være bedre end 150 meter (Alizadeh 2008).

GPS: Ved hjælp at en indbygget GPS-antenne vil stort set alle nyere smartphones kunne beregne deres position med hjælp fra satellitter. Med denne metode kan der opnås en nøjagtighed på omkring 10 meter (Alizadeh 2008), hvis forholdende for modtagelse af GPS-signalet er nogenlunde fornuftigt. Teknologien er dog både tids- og energikrævende og afhængigt af omgivelserne (det kan være et problem at få et ordenligt signal i eksempelvis gader med høje huse, under tæt beplantning og indendørs).

A-GPS: Er en videreudvikling af GPS-teknologien, se Figur 13. Den primære forskel ligger i, at dele af processeringen foregår på en assisterende server som har både mere regnekraft og information til positionsberegningen tilgængelig. Derved opnås en betydelig hurtigere bestemmelse af den faktiske lokation, batteriforbruget mindskes og nøjagtigheden øges en smule. Denne teknologi kræver dog, at enheden kan udveksle data med serverne via en form for netværk (Wang *et al* n.d).



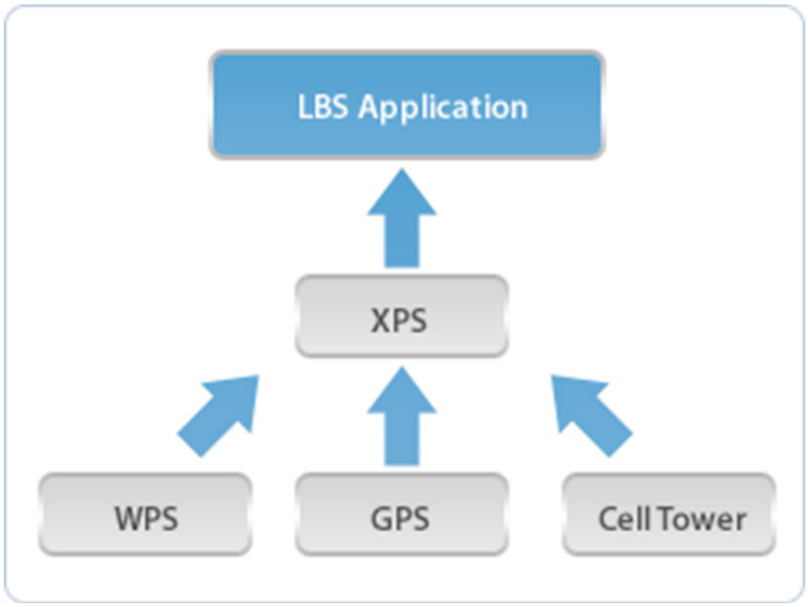
Figur 13: Forskel mellem GPS og A-GPS

Positionering vha. Wi-Fi (WPS): På samme måde som det var tilfældet ved sendemasterne, kan en lokation også bestemmes på baggrund af styrken af signalet af de tilgængelige trådløse WiFi-netværk. Services udbydes af en række selskaber (som eksempelvis Skyhook), som ligger inde med store databaser indeholdende information om de trådløse netværks MAC-adresser og deres fysiske placering (Zandbergen 2009). Smartphonen sender data om signalstyrken på de tilgængelige trådløse netværk videre til en server, som så sammenholder informationen med oplysningerne fra en database og herefter beregner smartphones lokation (Michal 2011).

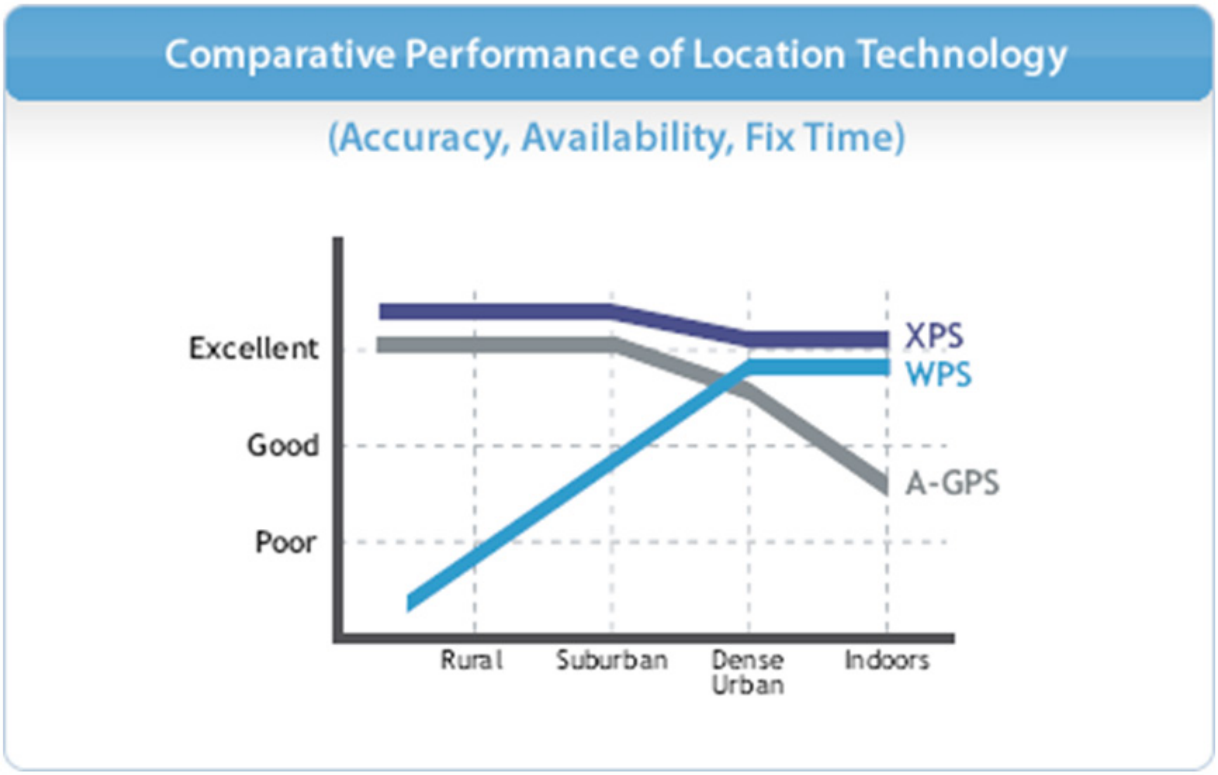
Nøjagtigheden i denne metode afhænger meget af antallet af tilgængelige netværk. I bymæssig bebyggelse vil der ofte være rigeligt med tilgængelige netværk, og derfor en god nøjagtighed, mens metoden i øde landområder ikke vil kunne benyttes. Under gode forhold kan der forventes en nøjagtighed på omkring 20 meter (Zandbergen 2009) og benyttes både til smartphone men også til lokationsbestemmelse af eksempelvis laptops (som vi kender det fra eksempelvis Google maps m.fl.).

Hybrid positioneringssystemer (XPS): Ovenstående metoder til lokationsbestemmelse har alle deres fordele og ulemper med hensyn til tids- og strømforbrug, nøjagtighed og tilgængelighed. For at få den bedste og hurtigste bestemmelse af en lokation kombineres de førnævnte metoder derfor ofte nogle hybridsystemer, hvor forcerne ved de forskellige systemer udnyttes. Metoden resulterer i et

system, som hurtigt og rimeligt nøjagtigt kan fastslå en given position. Nedenstående Figur 14, viser hvordan hybrid positioneringssystemer benytter flere forskellige teknologier til lokalisering. Figur 15 giver et billede af hvor god en præcisision der kan forventes, af de forskellige teknologier.



Figur 14: Diagram over XPS (Skyhook 2012)



Figur 15: Sammenligning af lokations teknologier (Skyhook 2012)

5.10 Kort projektioner

Et kort er en forsimplet model af virkeligheden, som afbilleder en del af jordens overflade. Kortets afbildning af virkeligheden og dets egenskaber afhænger af den valgte kortprojektion (Fajstrup 2006). Det er derfor vigtigt at være bevidst om kortprojektioners betydning når data skal præsenteres geografisk. I det følgende afsnit vil baggrunden og brug af kortprojektioner kort blive gennemgået.

Der vil altid opstå problemer når jordens krumme overflade søges afbilledet i planen. Der findes en lang række matematiske modeller (kortprojektioner), som på forskellig vis kan benyttes til dette. Modellerne kan dog kun opfylde nogle af de krav, der er til gengivelsen af virkeligheden. Nogle projektioner kan lave en arealtro afbildning af virkeligheden, mens andre bibeholder virkelighedens vinkler (en såkaldt konform projektion). Fælles for dem alle er dog, at en kortprojektion altid vil introducerer en hvis afstandsforvrængning (Microsoft 2012)

5.10.1 Web-mercator

Ligesom UTM-projektionen stort set er standard i Danmark, er den såkaldte Web-mercator-projektion standard på de største kortvisningstjenester på internettet (Google maps, Bing maps m.fl.). Web-mercatorprojektionen benytter sfæriske formler til beregning, hvilket resulterer i væsentlig simplere, og dermed hurtigere, beregning. Forsimplingen af beregningen medfører en forringelse af afbildningens konformitet, som dog må betragtes som tilladelig i forhold til kortets anvendelse.

Web-mercator-projektionen er implementeret i mange af de mest benyttede GIS-værktøjer, men understøttes ikke i data fra eksempelvis KMS. Beregningen til omprojektsering af KMS-data er dog så simple, at reelt ikke har den store betydning i GIS-sammenhæng (Kokkendorff 2010)

5.10.2 Håndtering af projektioner

Når der arbejdes med geodata, er det vigtigt at holde styr på projektionen fra data og det medie det skal afbilledes på. Stemmer disse ikke overens skal der ske en omprojektion. Ellers kan det resultere i et ubrugeligt grundlag for både analyser og illustration.

Heldigvis er der stor hjælp at hente i både rummelige databaser og visualiseringsværktøjer som eksempelvis GeoServer. Her er det nemlig muligt, at fortælle programmet hvilken projektion inputdata er i og hvilken projektion data ønskes outputtet i. Programmet sørger herefter selv for at omprojkterer data, inden det videreformidles. Omprojektering af store datamængder er selvfølgelig ressourcekrævende og det kan derfor med fordel overvejes om data skal omprojekteres inden det lagres i eksempelvis en database.

5.11 Userbility

Touchbaserede brugergrænsefalder(***User Interfaces*** - UI) er i stigende grad blevet standarden for smartphones og tablets. At kunne manipulere information på skærmen direkte med fingrene, giver brugeren en stærkere følelse af at være i kontrol over systemet, frem for at systemet dikterer arbejdsgangen. Et andet aspekt ved direkte interaktion, er at det kan gøre produktet tilgængelig for et bredere spektrum af brugere. Kombineret med et godt visuelt og velfungerende interface, kan touch forbedre den overordnede brugeroplevelse.

Der er dog en række ulemper, der må adresseres, når der designes til et touch UI. Et UI designet til mus og keyboard interaktion, kan ikke uden videre overføres, men må først redesignes til at fungere med touch.

5.11.1 Taktil respons

Til forskel fra mus og keyboard, har en touch skærm ingen taktil feedback, der indikerer over for brugeren, hvornår en knap er blevet trykket. Det kan derfor være nødvendigt at gøre brug af forskellige visuelle - og auditive feedback mekanismer, for at gøre det klart. Undersøgelser har, ifl. Bachl *et al* 2010, vist at taktil feedback kan forbedre præstationen og mindsker fejl.

5.11.2 Blokeret syn

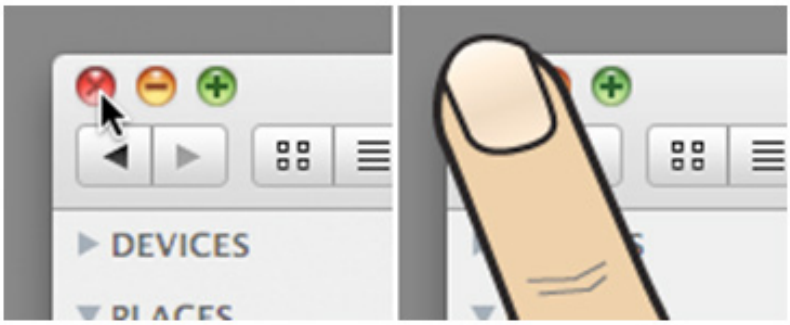
Når et touch UI designes, er det vigtigt at være opmærksom på, hvordan fingre og hænder kan bloker for skærmen. I modsætning til en mus, der altid kun optaget en meget lille portion af skærmen, kan fingre dække store dele af skærmen, og gøre det svært for brugeren at se hvad der foregår. Dette er i særlig grad udtalt på små skærme, og bør tages i med overvejelserne når UI elementer placeres.(Bachl *et al* 2010)

Et eksempel på en kreativ løsning på blokeret syn til skærmen, er keyboardsystemet på Apples iOS, hvor den aktive knap forstørres op, så brugeren kan se hvilken tast der er ramt, se Figur 16.

Figur 16: iOS keybord (BindApple 2008)

5.11.3 Præcision

Mennesker har meget forskellige hænder, og hvordan vi interagere med et touch UI, er til dels afhængigt af dette. Det er derfor nødvendigt at tage disse forskellige fysiske karakteristika med i overvejelserne, når et touch UI designes. Fx bør et touch objekt iflg. Wang & Ren 2009 ikke være mindre end 11,5mm. Det er ligeledes værd at overveje, hvordan fingrene på samme hånd er forskellig mht. størrelse og egenskaber. Fx benytter vi ikke tommelfingre og pegefingre på samme måde.



Figur 17: Nøjagtighed og blokerende effekt for hhv en musemarkør (th) og en finger (tv). (Bachl *et al* 2010)

En anden vigtig faktor at overveje er, hvor nøjagtige fingre er set i forhold til en musemarkør. En musemarkør har et pegefelt på én pixel, mens det er næsten umuligt at ramme en specifik pixel med en finger, se Figur 17.

5.11.4 Multi-touch

Multi-touch er et begreb, der dækker over systemer, der tillader mere end et input af gangen. Hvis et system understøtter multi-touch, er det muligt også at undersøtte mere avancerede gestikuleringer, som fx at to fingre fra hinanden zoomer ind eller at fire fingre henover skærmen skifter skærmbillede.

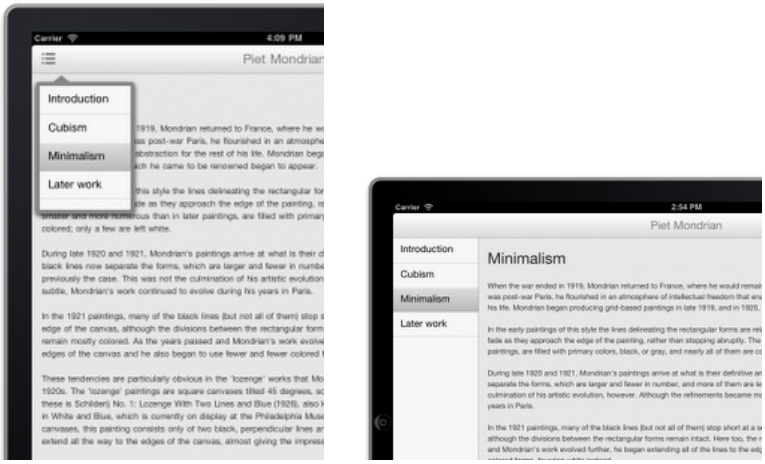
Jo mere avancerede gestikuleringerne bliver, des færre brugere vil kunne udføre dem, uden instruktion. Bla. derfor bør kompleksiteten af gestikuleringen matche kompleksiteten af den pågældende handling.(Bachl *et al* 2010)

5.11.5 Visning

For så godt som alle mobileapplikationer(app) glæder det, at der kun er ét vindue på skærmen af gangen, som optager hele skærmen. Det er således ikke muligt at rykke rundt på applikationen, eller se andre applikationer på samme tid. Dette er fundamentalt forskelligt fra et klassisk computersystem (fx Microsoft Windows eller Apple OS), hvor brugeren ofte har flere programmer på skærmen af gangen. Forskellen er vigtig at holde sig for øje når et mobil touch UI designes, da det er en del af det der giver apps et særligt fokuseret udtryk.

En anden fundamental forskel mellem skrivebordscomputere og mobil enheder, er muligheden for at dreje skærmen. Langt de fleste moderne smartphones og tablets har indbygget gyrometer, hvilket gør dem i stand til at ændre layout efter orientering. Dette bør tages med i overvejelserne når UI designes, da det skal kunne fungere både i portræt- og landskabsorientering (Apple Inc. 2012). Skærmstørrelse og orientering er måske de to største udfordringer, når der designes UI til mobileenheder, og det er essentielt, at der gøres overvejelser om dette. Fx kan grafiske elementer der fungere godt på en stor skærm i landskabsorientering, fungere mindre godt på en lille skærm i portrætorientering. Figur 18 og Figur 19 er et eksempel på en app, designet til at skifte layout, efter hvilken enhed og hvilken orientering der benyttes.

Figur 18: Smartphone app med to layouts til hhv. portræt(tv.) og landskab(th.). (Pearce 2012)

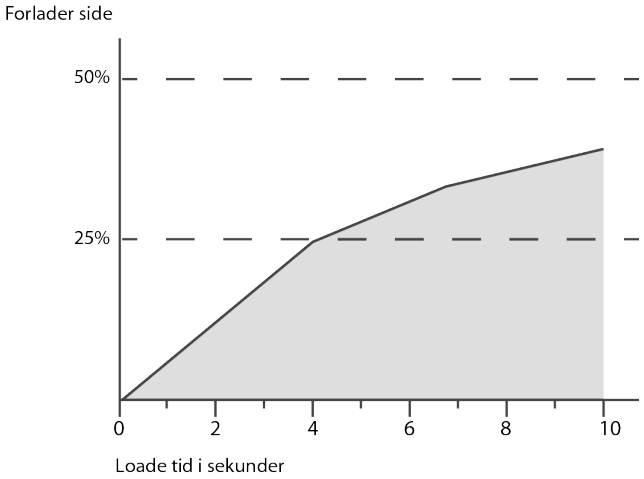


Figur 19: Samme applikation som på forgående billede, vist på en tablet. (Pearce 2012)

5.11.6 Hastighed og performance

Hvor hurtigt en webapplikation eller hjemmeside loader, er af afgørende betydning. Ifølge Google research vil 50 % af mobilebrugere forlade en side, hvis den ikke er loadet inden for 10 sekunder og 60 % af dem vil ikke komme tilbage til siden. (Google 2009). Disse tal er fra 2009, og i takt med at net hastigheden på mobileenheder bliver bedre og bedre, må det forventes at brugernes tålmodighed falder. Tal fra performance test firmaet Gomez.com viser, at 58 % af brugerne i 2009 forventede at en side loadede lige få hurtigt, eller hurtigere, på en mobilenhed som på en stationær computer. Dette tal var i 2011 steget til 71%.

Brugernes tålmodighed er tilmed betydeligt lavere på stationære computere, hvor 25 % forlader en side hvis den bruger mere end 4 sekunder på at loade, se Figur 20



Figur 20: Antal der forlader en side som effekt af loadetid (Google 2009)

5.11.7 Konsistens

UI design bør bestræbe sig på at være konsistent gennem helle app'en. Dvs. at knapper og ikoner gør det sammen, lige meget hvor man befinder sig i systemet. Et konsistent design gør det lettere for brugeren at navigere og benytte funktionerne smidigt og hurtigt. Konsistens bør også tænkes uden for den pågældende app, ved at der benyttes termologier og ikoner der er velkendte fra andre systemer. Det er her vigtigt, at ikoner gør hvad man forventer og kender fra andre apps, og ikke pludseligt gør noget andet. (Apple Inc. 2012)

5.12 Afrunding teori

I det foregående er teori om bl.a. databaser, applikationsformer, API'er og meget andet blevet gennemgået. På denne baggrund vil der i det følgende være en kort begrundelse af valgene foretaget i forbindelse med projektets implementering.

I afsnittet Lokale, Web- og hybride applikationer(5.8) redegjordes for fordele og ulemper i forbindelse med forskellige applikationstyper. Under hensyntagen til projektets længde, projektgruppens erfaring med kodning og ønsket om at udvikle en applikation, som ikke er operativsystemsafhængigt (Interoperabilitet), vælges der at udarbejde en Web-applikation. Derved udvikles en applikation, med et "native" -app-udseende, som hverken skal programmeres specielt til de forskellige styresystemer eller gennem godkendelsesproceduren i div. App stores. Til gengæld vil applikationen ikke have adgang til alle enhedens funktioner, ligesom hastigheden ved brug af applikationen, må forventes at være en smule langsommere end ved en lokal applikation.

Som nævnt i afsnit 5.11 *Userbility*, skal der gøres en del overvejelser i forbindelse med design af en touch-baseret brugergrænseflade. I dette projekt har vi valgt at benytte Sencha Touch som grundsetup til design af brugergrænseflade og håndtering af touch-funktionaliteter, da dette touch bibliotek er meget anerkendt på diverse udviklerfora. Valget af Sencha Touch betyder et fravalg i forhold til enheder, som ikke benytter sig af Webkit kompatible browsere. Det skyldes at Sencha Touch, teknologisk, er med helt fremme i skoende, og derfor benytter sig af de nyeste teknologier som eksempelvis HTML5 og CSS3, som ikke på nuværende tidspunkt er integreret i alle browsere. Dette opvejes dog i høj grad de muligheder der findes i Sencha Touch. Et stort og veldokumenteret bibliotek af standardfunktioner, layouts samt integrationsmuligheder med eksempelvis Google maps, er grunden til at Sencha Touch er valgt til dette projekt.

Sencha Touch benytter korttjenesten fra Google Map JavaScript API v.3 som standard. Google maps er en velkendt og hurtig korttjeneste, som tilbyder en del ekstra funktionaliteter i form af eksempelvis geokodning og værktøjer til opmåling og tegning. Både korttjeneste og tilhørende funktioner er veldokumenterede, og det er derfor valgt at benytte denne løsning i forbindelse med udarbejdelsen af dette projekt. Som nævnt i afsnit 5.5 *Korttjenester ift. Open cource, API og SDK* kan valget af Google medføre begrænsninger/betaling ved et stort antal daglige visninger. Denne begrænsning har dog ingen betydning for dette projekt, da brugen af applikationen, og dermed antallet af kortvisninger, vil være ganske begrænset.

Til datalagring benyttes en PostgreSQL som er et ORDBMS (se afsnit 5.4). Valget af dette DBMS grunder i PostgreSQL som et velrenommeret og pålideligt system, som med tilføjelsen af PostGIS giver den rummelige dimension(SDBMS), og tilhørende funktionalitet, som kræves for at løse de i projektet fastsatte opgaver (rummelige forespørgsler, indeksering mm). PostgreSQL/PostGIS er desuden en Open Source løsning, som frit kan benyttes og ikke stiller krav til betaling/licenser.

Til visualisering af de lagrede data benyttes en kortserver. Som nævnt i afsnit 5.2 *Data og datastandarder* findes der en række både Open Source og kommercielle løsninger. Da Open Source løsningerne kan dække projektets behov fuldt på højde med de kommercielle løsninger vælges disse.En undersøgelse mht. performance af to af de store Open Source kortservere, MapServer og GeoServer, viser ikke nævneværdige forskelle (Anderson & Deoliveira, 2007). På denne baggrund vælges der at arbejde videre med GeoServer, da dette er et værktøj som er blevet introduceret til os i undervisningen. Geoserver integreres let med PostGIS, tilbyder et tilgængeligt brugerinterface og mulighed for nem styling af de forskellige datalag. Geoserver tilbyder desuden cache-lagring af korttiles.

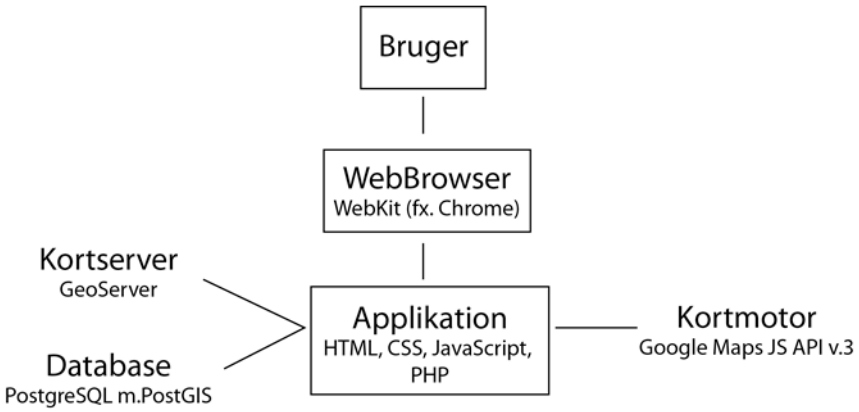
6. Implementering

I det følgende afsnit beskrives hvorledes en WebApp til offentlig digital forvaltning, er udarbejdet ud fra de ovenstående teorier og tekniker. En systematisk gennemgang af databaseopsætning, webserver, funktionaliteter og brugte API'er.

6.1 Systembeskrivelse

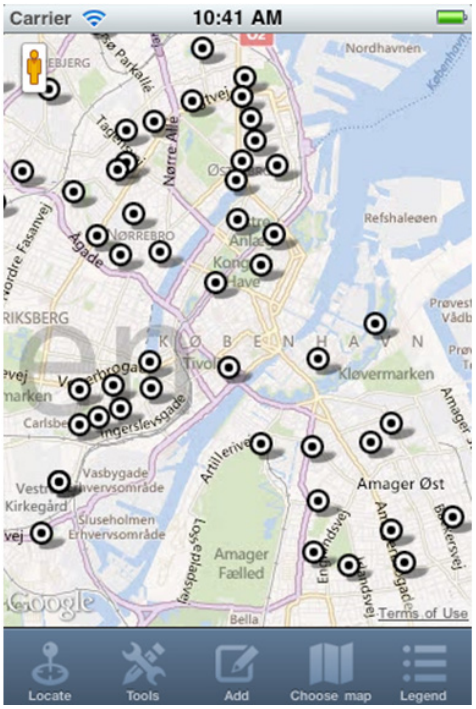
Systemet der udvikles, er en WebApp til håndtering af offentlige data; paragraf-3 områder, frede fortidsminder , skoler mv. Opbygningen af applikationen foregår inden for de rammer og begrænsninger, der defineres af brugerinterfacebiblioteket(Sencha Touch v.1.1), kort API'en (Google Map JavaScript API v.3) og div. Webkit baserede webbrowser.

Figur 21 viser det overordnede set-up. Brugeren interagerer med applikationen via. WebBrowseren på en desktop pc, en tablet eller en smartphone. Applikationen opbygges i HTML, CSS, JavaScript og PHP, og data håndteres af en PostgreSQL database i samarbejde med en GeoServer, der leverer WMS og KML overlays til fremvisning vha. Google Maps kortmotoren.



Figur 21: Applikationens overordnede set-up

Applikationen er opbygget omkring et centralt kort, der har hovedfokus gennem hele applikationen. I bunden af skærmen er placeret en toolbar, der indeholder fem undermenuer; "Locate", "Tools", "Add" , "Choose map" og "Legend", se figur Figur 22.



Figur 22: Applikationens startskærm

Undermenuen "Locate" har to måder, hvorved brugeren kan centrere kortet. Enten kan brugeren finde sin nuværende position vha. XPS, se afsnit 5.9 *Lokalisering*, eller brugeren kan centrere kortet omkring en ønsket adresse vha. en geokoder.

Undermenuen "Tools" indeholder adskillige værktøjer til rumlige kald og måling:

- "Near point" – Find information om et givent temalag, inden for en brugerdefineret radius af et valgt punkt.
- "Near line" – Find information om et givent temalag inden for en brugerdefineret radius af en indtegnet linje
- "Within Polygon" – Find information om et givent temalag inden for et brugerdefineret areal
- "Measure length" – Mål længde
- "Measure area" – Opmål areal
- "Get Coordinates" – Få koordinaterne for et givent punkt

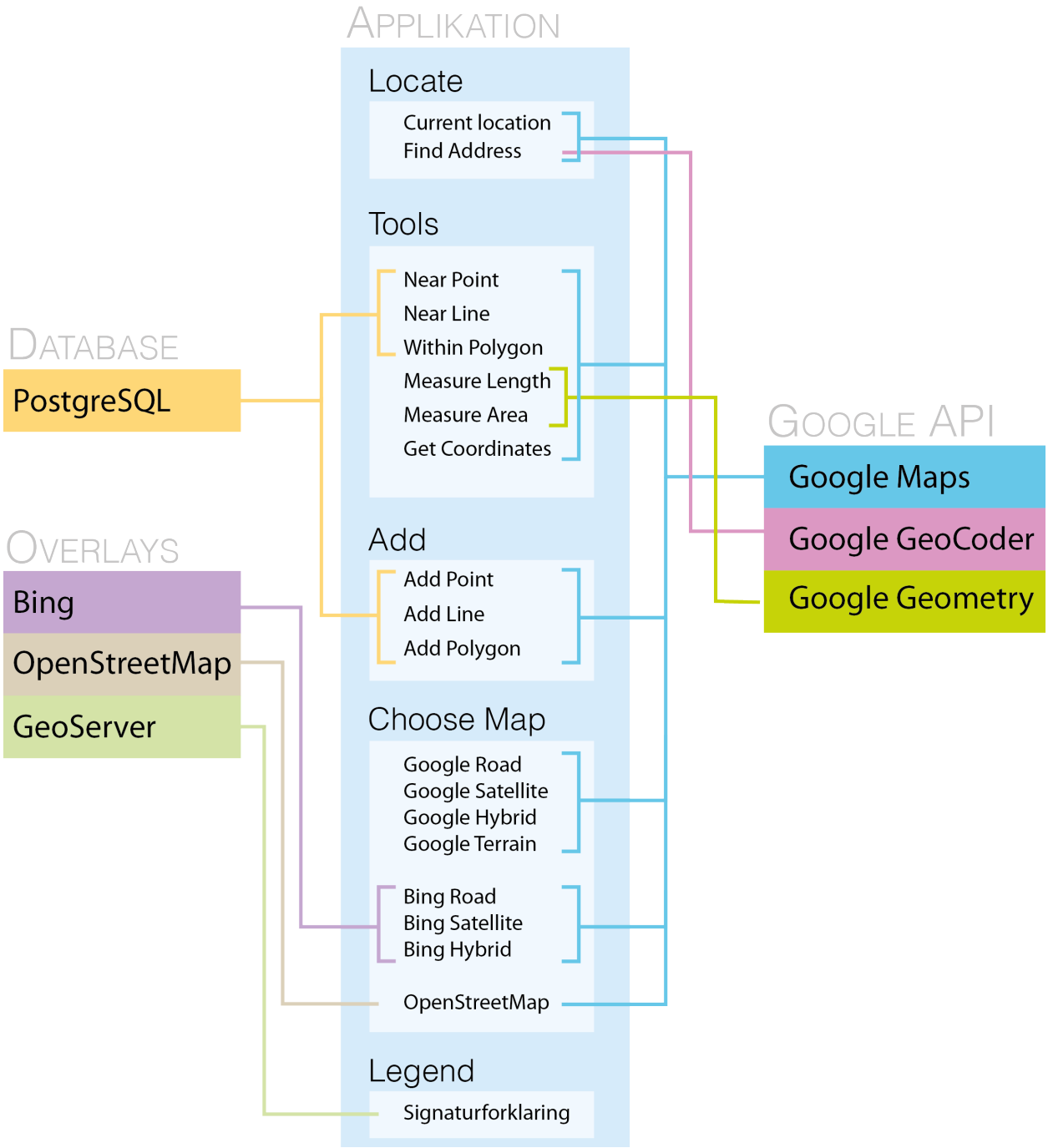
I undermenuen "Add" har brugeren mulighed for at oploade forskellige typer rumlig data til databasen vha. funktionerne:

- ”Add point” - Tilføjer et punkt til databasen
- ”Add line” – Tilføjer en linje til databasen
- ”Add polygon” – Tilføjer et polygon til databasen

”Choose map” undermenuen gør det muligt for brugeren, at skifte mellem flere forskellige baggrundskort, afhængigt af hvilket der passer bedst til en given opgave. Følgende baggrundskort er tilgængelige:

Google Road	Tematisk vejkort
Google Satallite	Satellit og luftfotos
Google Hybrid	Satellit og luftfotos i kombination med hovedveje
Google Terrain	Terræn kort
Bing Road	Tematisk vejkort
Bing Satellite	Satellit og luftfotos
Bing Hybrid	Satellit og luftfotos i kombination med hovedveje
Open Street Map	Open Source wiki kort

”Legende” knappen, yderst til højre på toolbaren, skifter kortet ud med en signaturforklaring over de viste kortoverlays fra GeoServeren.



Figur 23: Diagram over applikationens funktioner og deres forbindelser

Figur 23 viser alle applikationens funktionaliteter, samt hvilke forbindelser de hver især har til applikationens forskellige dele. I de følgende afsnit vil funktionerne, og hvordan de er opbygget, blive beskrevet nærmere.

6.2 Framework

Blandt de grundlæggende forskelle på Apps og internetsider er hvor fokuserede Apps er. En App optager hele skærmen og forstyrrende elementer, som webbrowserensadressebar og andre programmer, er skubbet i baggrunden. Ligeledes har en god App et intuitivt interface som, en erfaren App-bruger vil kunne benytte uden hjælp. Ønsker man at en WebApp skal opnå det samme fokuserede udtryk, er det nødvendigt at designe WepApp'en derefter.

Til dette formål gør vi brug et programmeringsbibliotek kaldet Sencha Touch. Sencha Touch er et HTML5 mobil applikations framework opbygget omkring JavaScript. Det vil sige al udviklingen med Sencha Touch foregår vha. JavaScript, men flere af de funktioner og effekter Sencha Touch biblioteket indeholder, bygger på HTML5- og CSS3 teknologier. Fordelen ved at opbygge en WebApp omkring et sådan framework er, at det håndterer flere af de layoutmæssige ting, der ellers ville tage meget lang tid at kode fra bunden. Ligeledes er der flere funktioner der gentages adskillige gange i genneme WebApp'en, som man med et framework, ikke behøver gentage, men blot kan kaldes fra biblioteket.

At skabe et "app-layout" i Sencha Touch er forholdsvis simpelt, men det kræver at der tænkes lidt anderledes, ifht. traditionel websideudvikling. Kodebid 22 viser "index.html" i vores set-up (se evt. Bilag 1).

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Alfred</title>

        <link rel="stylesheet" href="touch/resources/css/sencha-touch.css"
type="text/css">
    <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=tru
e&libraries=geometry"></script>
        <script type="text/javascript" src="touch/sencha-touch.js"></script>
        <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/
jquery/1.7.1/jquery.min.js"></script>
        <script type="text/javascript" src="js/slider.js"></script>
        <script type="text/javascript" src="js/funktioner.js"></script>
        <script type="text/javascript" src="js/index.js"></script>

</head>
<body>
</body>
</html>
```

Kodebid 22: index.html

Som det ses indeholder "index.html" ikke andet end en lang række henvisninger til CSS, Google API'er og JavaScripts. Gennemgår vi dokumentet fra toppen er det som følger.

```
<!DOCTYPE html>
<html>
<head>
...
</html>
```

Kodebid 23: Doctype og header

Kodebid 23 fortæller browseren, at der er tale om et HTML5 dokument via `<!DOCTYPE html>`, mens `<html>` og `</html>` tagene angiver hvor html dokumentets indhold starter og slutter.

```
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Alfred</title>
    ...
</head>
<body></body>
```

Kodebid 24: meta og title

Et HTML5 dokument skal, iflg. standarden, indeholde et `<head>`- og et `<body>`-element, se Kodebid 24. Bemærk at `<body>`-elementet i vores tilfælde er tomt. Da indhold og brugerinterface vil blive genereret af vores JavaScript. `<Head>`-elementet er dog nød til at indeholde et minimum af metadata(`<meta>`) om dokumentets indhold ("content type") og bogstavesæt ("character set") . Bemærk at vi benytter "utf-8", der tillader nordiske bogstaver(Æ, Ø og Å).

`<title>` angiver sidens titel, og fremgår dels i toppen af browservinduet samt som applikationens predefinerede navn når "Save to home screen" udføres på en iOS enhed. Titlen "Alfred" er en sammentrækning af Alsing og Fredskild.

Endelig indeholder `<Head>`-elementet de førnævnte henvisninger til div. JavaScripts, kort API'er og CSS, se Kodebid 25

```
<link rel="stylesheet" href="touch/resources/css/sencha-touch.css" type="text/css">
<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=true&li
braries=geometry"></script>
<script type="text/javascript" src="touch/sencha-touch.js"></script>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/
jquery/1.7.1/jquery.min.js"></script>
<script type="text/javascript" src="js/slider.js"></script>
<script type="text/javascript" src="js/funktioner.js"></script>
<script type="text/javascript" src="js/index.js"></script>
```

Kodebid 25: Henvisninger til CSS, JavaScripts og API

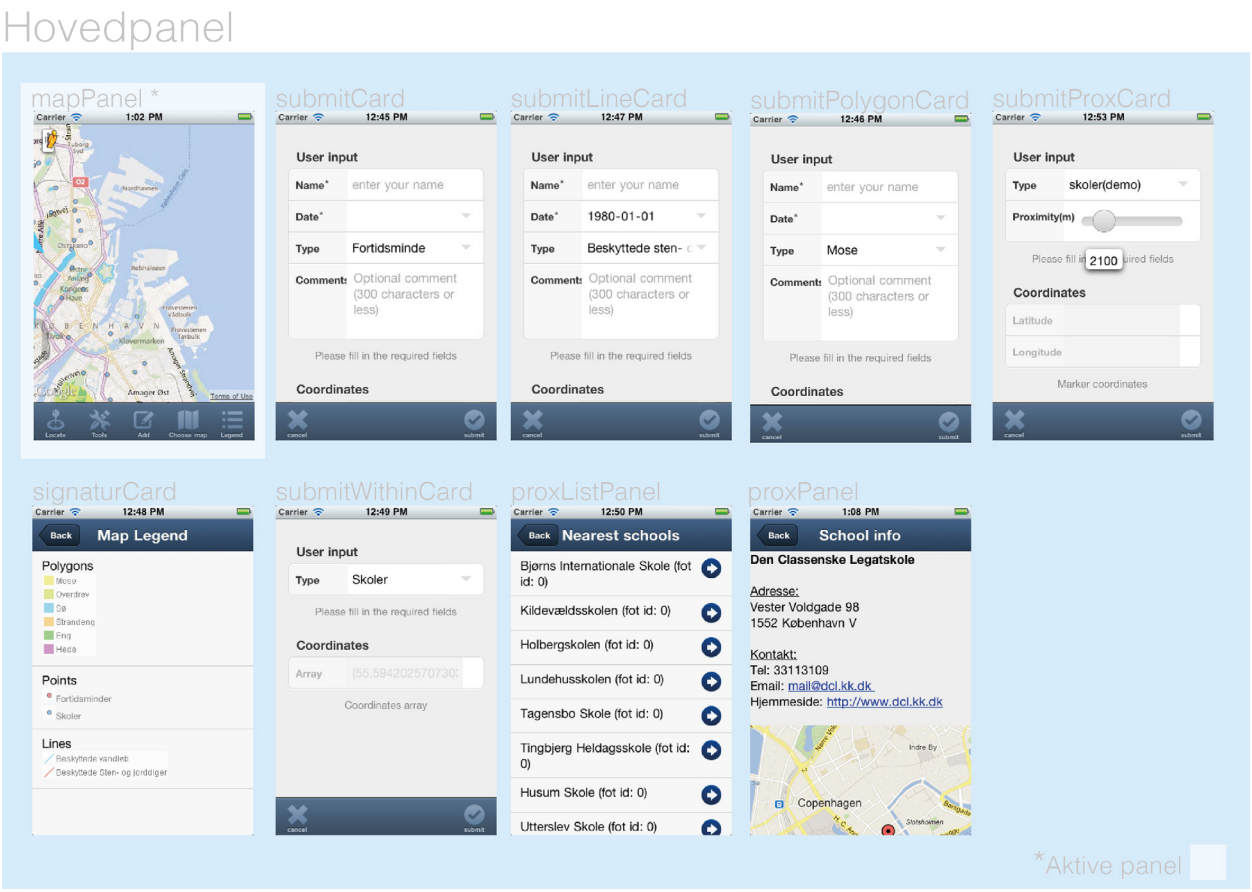
Taget fra toppen henviser Kodebid 25 til Sencha Touchs CSS, Google Maps API m. geometry underbiblioteket, Sencha Touch biblioteket, Jquery biblioteket(et andet JavaScript bibliotek der også benyttes i applikationen), Slider.js (et mindre script til at styre en slider funktionalitet) og endelig funktioner.js og indeks.js, der indeholder selve vores applikation.

I "index.js" har vi placeret applikationens struktur, toolbars, panels, knapper osv. Figur 24 viser hvordam disse giver sig ud i den færdige applikation.

Figur 24: Panels(1), List(2), Toolbar(3)

En Sencha Touch applikation er opbygget af paneler, se ovenstående Figur 24, der kan indeholde flere underliggende paneler. Et panel kan indeholde almindelige HTML elementer, samt Sencha Touch elementer som fx toolbars, datepickers, carousels, lists mm.

I vores applikation arbejdes der med et hovedpanel(kaldet: ”panel”) og adskillige underpaneler, der på skift optager hele hovedpanelet, se Figur 25.



Figur 25: Diverse underpanel der er på skift optager hele hovedpanelet

Kodebid 26 viser hvordan hovedpanelet(”panel”) bliver defineret, se også bilag 2 for den samlede kode i *index.js*.

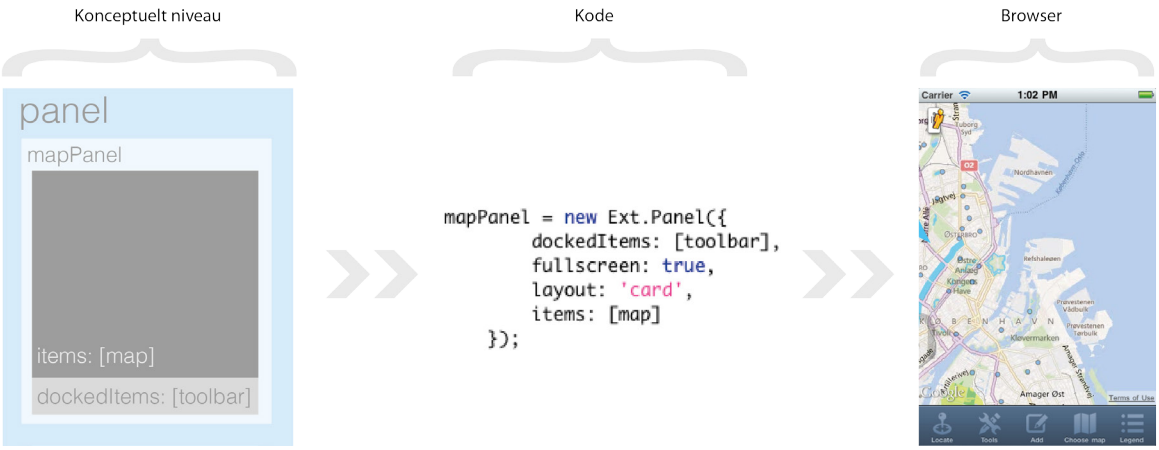
```
Panel = new Ext.Panel({
            dockedItems: [infoBar, infoBar1,  ],
            fullscreen: true,
            layout: 'card',
            cardSwitchAnimation: 'slide',
            items: [mapPanel, signaturCard,... ]
      });
```

Kodebid 26: Hovedpanelet

Som det ses, er der tale om et Sencha Touch element, defineret af kommandoen "`new Ext.Panel`" med følgende parameter:

- `dockedItems` – Henviser til infobar og infobar1, som er defineret et andet sted i index.js
- `Fullscreen: true` – Fortæller Sencha Touch at elementet skal optage hele skærmen
- `Layout: 'card'` – Angiver hvordan underpaneler håndteres ift. Hovedpanelets dimensioner.
- `cardSwitchAnimation: 'slide'` – Angiver hvordan den animerede overgang mellem underpanelerne skal foregå
- `item: [mapPanel, signaturCard, sumbitCard ...]` – Angiver hvilke underpaneler hovedpanelet har.

Når applikationen startes op, vil det aktive underpanel altid være 'mapPanel'. MapPanel indeholder baggrundskortet, samt værktøjslinjen "toolbar", der er et dockedItem, se Figur 26.



Figur 26: Diagram over applikationen på konceptuelle-, kodede- og browserniveau

Denne opbygning gentager sig gennem hele applikationen, således at alle underpaneler er defineret med deres egen toolbar (dockedItems) og indhold (items). Den "toolbar" der benyttes i "mapPanel" genbruges på flere underpaneler, i en mere eller mindre modificeret udgave. Kodebid 27 viser hvordan "toolbar" er opbygget.

```
var toolbar = new Ext.TabBar({
      dock : 'bottom',
      id: 'toolbarId',
      ui   : 'light',
      hidden: false,
      layout: {pack: 'center'},
      items: [
            {
                  iconMask: true,
                  text: 'Locate',
                  title: 'locate',
                  id: 'locateid',
                  iconCls: 'locate',
                  handler: ExtMap.UI.showLocateListOverlay,
                  },

...

                  {
                  iconMask: true,
                  text: 'submit',
                  id: 'submitLineId',
                  title: 'submit',
                  iconCls: 'check_black2',
                  hidden: true,
                  handler: submitLineCardHandler,
                  }
      ]
});
```

Kodebid 27: Toolbar

Som det ses er toolbar defineret som en variabel af typen "`Ext.TabBar`", med følgende parameter:

`dock: 'bottom'` – angiver at baren skal placeres i bunden af panelet

`id: 'toolbarID'` – angiver et unikt ID til variablen

`ui: 'light'` – angiver at vi ønsker at arbejde med Sencha Touchs lyse standard layout

`hidden: false` – benyttes til at gemme toolbaren

`layout: {pack: 'center'}` – angiver at knapperne skal placeres centreret

`items: [...]` – indeholder knapperne

Kodebid 28 viser et eksempel, på hvordan en af toolbarens knapper eller TabBar.items er kodet:

```
{
iconMask: true,
text: 'submit',
id: 'submitLineId',
title: 'submit',
iconCls: 'check_black2',
hidden: true,
handler: submitLineCardHandler,
}
```

Kodebid 28: Toolbar item

For hvert TabBar.item er angivet et unikt id(`id`), en titel(`title`) og den tekst der står under ikonet (`text`). Ligeledes er angivet et ikon(`iconCls`) og hvordan det givne ikonet vises (iconMask er en style til ikoner i Sencha Touch). Endelig er det angivet, hvad der sker når knappen trykkes, ved at tilknytte en handler. Handleren er en særlig type henvisning i Sencha Touch, som kører en funktion et andet sted i koden, når knappen trykkes. I dette eksempel henvises til funktionen `submitCardHandler`, se Kodebid 29, der er en funktion til at skifte det aktive panel.

```
var submitCardHandler = function(button, event) {
                    Panel.setActiveItem(submitCard, {type: 'slide'});
                    submitForm.setValues({lat: markerLat, lng:
markerLng});
};
```

Kodebid 29: Submit handler

Sencha Touch kommer med fire pre-compilede CSS dokumenter(sencha-touch.css, apple.css, android.css og bb6.css), se Figur 27. Som bekendt benytter vi i vores applikation en version af *sencha-touch.css*, se første linje i Kodebid 25. Der er ikke foretaget nogen nævneværdige ændringer til dette CSS ift. udsende, men der er tilføjet et par ikoner, og fjernet et par overflødige "style elementer".

Figur 27: Sencha-touch.css, bb6.css, apple.css og andriod.css

Kodebid 30, viser de ændringer, vi har foretaget til standard *sencha-touch.sass* dokumentet, inden det compiles til det endelige CSS dokument. De med rødt markerede elementer, er elementer der er fjernet fra standard layoutet, og det med grønt markerede elementer er tilføjelser.

```
@import 'sencha-touch/default/all';

@include sencha-panel;
@include sencha-buttons;
@include sencha-sheet;
@include sencha-picker;
@include sencha-tabs;
@include sencha-toolbar;
@include sencha-toolbar-forms;
@include sencha-carousel;
@include sencha-indexbar;
@include sencha-list;
@include sencha-list-paging;
@include sencha-list-pullrefresh;
@include sencha-layout;
@include sencha-form;
@include sencha-msgbox;
@include sencha-loading-spinner;
@include pictos-iconmask('trash');
@include pictos-iconmask('check_black2');
@include pictos-iconmask('settings3');
@include pictos-iconmask('settings5');
@include pictos-iconmask('home');
@include pictos-iconmask('list');
```

Kodebid 30: Pre compliling af SASS til Sencha Touch

Ovenstående compiles via Compass til det CSS dokument, der benyttes i vores applikation. Som det ses er de eneste tilføjelser til standardopsætningen de seks ikoner i bunden af SASS dokumentet. Det ville have været muligt at foretage betydeligt mere omfattende ændringer i udsende af vores applikation vha. SASS, men bl.a. pga. tidsrammen blev dette ikke prioriteret.

Med vores applikationsdesign har vi bestræbt vi os på at lave et touch brugerinterface, med store fingervenlige knapper, under hensyntagen til elementerne beskrevet i teoriafsnit 5.11 *Usability*. Ligeledes er der gjort overvejelser om værktøjslinjens placering ift. kortindholdet. Placeringen i bunden af applikationen gør det muligt, at tilgå flere af applikationen funktioner med tommelfingeren, uden at spærre for udsynet til kortet se Figur 24. Yderligere er de mange paneler, der på skift optager hele skærmbilledet, med til at sikre den meget begrænsede skærmstørrelse ikke bliver for overfyldt.

6.3 Database set-up

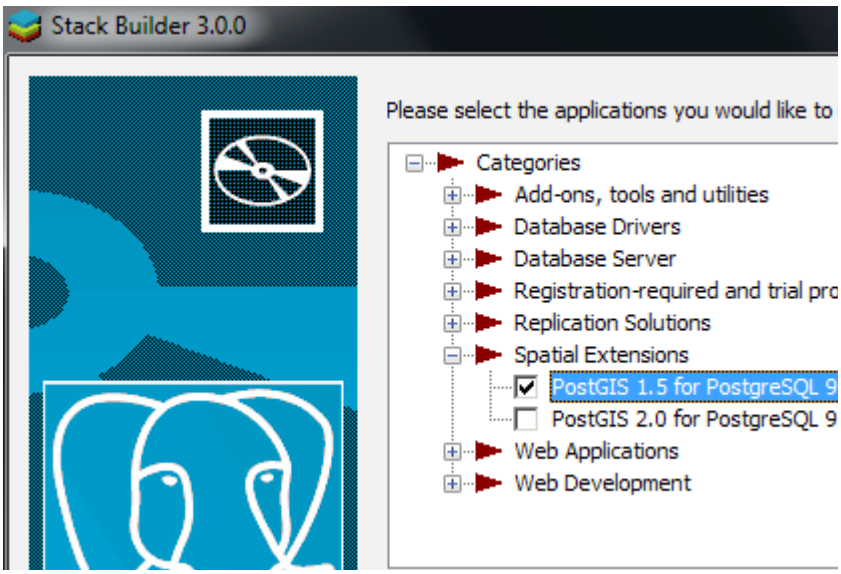
I det følgende afsnit gennemgås opsætningen af det i projektet benyttede database set-up, herunder oprettelse af tabeller og indlæsning af data.

6.3.1 Installation af PostgreSQL og PostGIS

Som nævnt, i afsnit 5.12 *Afrunding teori,* vælges der i dette projekt at arbejde med en *PostgreSQL* database med den rummelig overbygning *PostGIS*.

Via siden www.enterprisedb.com/products-services-training/pgdownload#windows hentes en PostgreSQL-installationsfil, som foruden selve databaseserveren også indeholder programmer som *pgAdminIII* og *Stackbuilder* (3.0.0), der benyttes senere i projektet. Til brug i projektet hentes en installationspakke for PostgreSQL 9.0 (Win x86-32), som installeres med hjælp fra den medfølgende guide.

Med databasen installeret er det nu tid til at tilføje den rummelige overbygning PostGIS. Dette gøres nemt ved hjælp af programmet Stackbuilder, hvor PostGIS (version 1.5) vælges under "Spartial Extensions", se Figur 28.



Figur 28: Stackbuilder installations vinduet

Endeligt installeres plug-in'et *shp2pgsql* til pgAdminIII, som muliggøre import af shapefiler til databasen. Plug-in'et hentes på www.postgis.org/download/windows/ og installeres som beskrevet i den medfølgende vejledning.

Oprettelse og konfiguration af den egentlig database foregår gennem programmet pgAdminIII, som, vha en grafisk brugerflade, administrer databaseserveren. Via guiden i pgAdminIII oprettes der en database med navnet geoDB på baggrund af den prædefinerede PostGIS-template. Databasen er nu klar til af få tilføjet data. Dette sker også gennem pgAdminIII.

6.3.2 Data

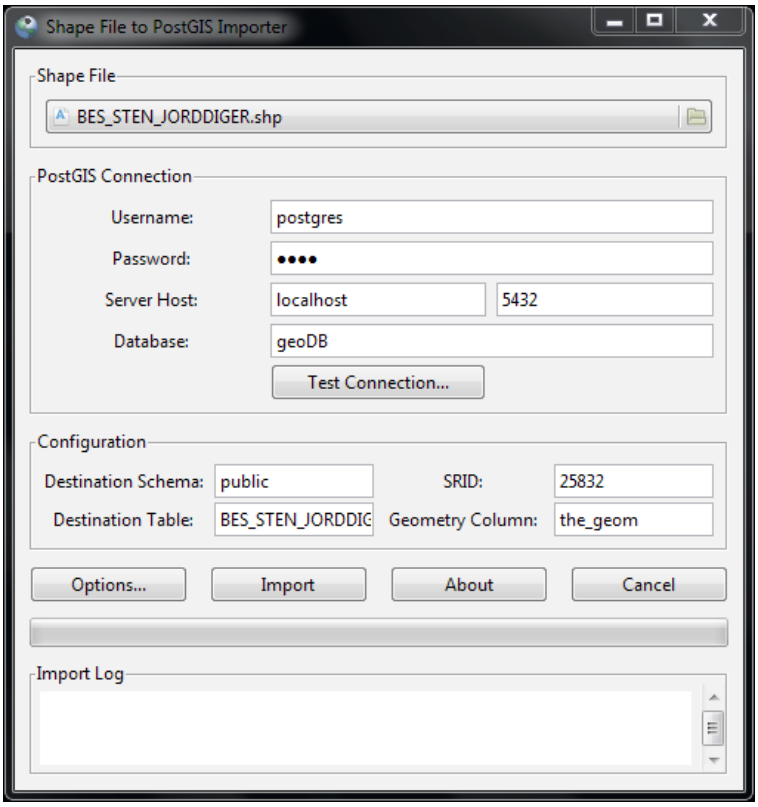
I forbindelse med projektudarbejdelsen bruges paragraf-3 data, som case for udviklingen af en applikation til brug i digital forvaltning. Det er vigtigt, at data i applikationen repræsenteres som både punkter, linjer og flader, for at demonstre at løsningen kan håndtere alle disse datatyper. Udover naturdata, benyttes også et datasæt om Københavns skoler. Dette datasæt er medtaget, for at kunne demonstrere de mange udnyttelsesmuligheder, der er i forbindelse med brug af attributter. Applikationen benytter således følgende data, se Tabel 4:

Data	Type
Beskyttede sten- og jorddiger	Linje
Beskyttede vandløb	Linje
Eng (§3-tabel)	Flade
Fredede fortidsminder	Punkt
Hede (§3-tabel)	Flade
Mose (§3-tabel)	Flade
Overdrev (§3-tabel)	Flade
Skoler (kun København)	Punkt
Strandeng (§3-tabel)	Flade
Sø (§3-tabel)	Flade

Tabel 4: Applikations data

Paragraf-3 data er hentet fra *arealinfo.dk*, ved hjælp af tjenestens downloadfunktion, som en ESRI shape-fil i referencesystemet UTM 32N EUREF89 (epsg: 25832). Skoledata er fundet online og opsat som CSV-fil, indeholdende information om de enkelte skoler, heriblandt koordinater for skolens placering. CSV- filen er vha. ESRI ArcMap konverteret til en shape-fil i referencesystemet UTM 32N EUREF89.

Shape-filerne tilføjes databasen ved hjælp af pgAdminIII-plug-in'et shp2pgsql. Plug-in'et kræver en shape-fil som input, samt oplysninger til databaseadgang og shape-filens referencesystem (SRID), se Figur 29.



Figur 29: shp2pgsql plugin

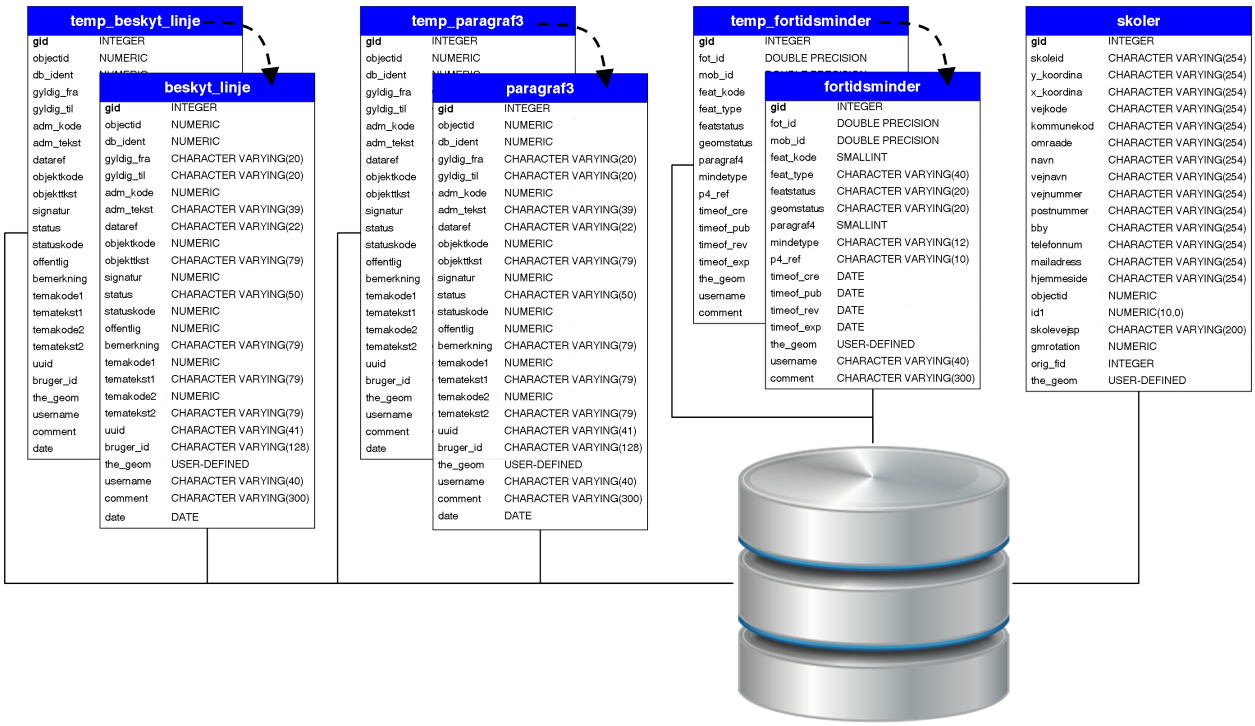
Data fra shape-filen indlæses herefter i den definerede tabel (*Destination Table*) i databasen. I løbet af denne proces sker der også en indeksering af data. Dette sker for at optimere databasens ydelse. Som standard benytter PostGIS en GiST-indeksering.

6.3.3 Databaseopsætning mm.

Efter indlæsning af ovenstående shape-filer indeholder databasen nu tabellerne; *beskyt_linje*, *paragraf3*, *fortidsminder* og *skoler*. Dette er imidlertid ikke nok til at understøtte projektapplikationens opbygning. Projektapplikationen stiller krav til både at kunne få vist databasens indhold som et overlay på et kort, samt at brugeren kan tilføje data og ikke mindst se det tilføjede data på kortet umiddelbart efter. For at opnå den bedst mulige performance, vil datavisningen foregå ved hjælp af prægenererede tiles – mere om dette i afsnit 6.4 *Web- og GeoServer set-up*. Dette medfører, at tilføjet data ikke vil vises for brugeren, før de berørte tiles regenereres, hvilket kan være en tidskrævende proces.

For at afhjælpe dette problem indberettes data fra applikationen til en midlertidig tabel, med samme struktur som originaltabellen. Data fra den midlertidige tabel, vil herefter kunne vises for brugeren med det samme vha. et KML-overlay – mere om dette i afsnit 6.5 *Kortimplementering*. Et serverscript vil herefter kunne sørge for at flytte data fra de midlertidige tabeller til de tilhørende grundtabeller, samt regenererer tiles. Dette serverscript er dog ikke en del af dette projekt.

Nedenstående figur Figur 30 viser hvordan data lagres i tabeller i projektdatabasen.



Figur 30: Databaseopbygning

De midlertidige tabeller oprettes via pgAdminIII og med brug af følgende SQL-kode, se Kodebid 31

```
CREATE TABLE temp_fortidsminder
(
  gid integer,
  fot_id double precision,
  mob_id double precision,
  feat_kode smallint,
  feat_type character varying(40),
  featstatus character varying(20),
  geomstatus character varying(20),
  paragraf4 smallint,
  mindetype character varying(12),
  p4_ref character varying(10),
  timeof_cre date,
  timeof_pub date,
  timeof_rev date,
  timeof_exp date,
  the_geom geometry,
  username character varying(40),
  "comment" character varying(300),
  CONSTRAINT temp_fortidsminder_pkey PRIMARY KEY (gid),
  CONSTRAINT enforce_dims_the_geom CHECK (st_ndims(the_geom) = 2),
  CONSTRAINT enforce_geotype_the_geom CHECK (geometrytype(the_geom)
= 'POINT'::text OR the_geom IS NULL),
  CONSTRAINT enforce_srid_the_geom CHECK (st_srid(the_geom) = 25832)
)
```

Kodebid 31: SQL-kode til oprettelse af tabel

Foruden oprettelsen af kolonner angiver SQL-koden følgende bindinger til tabellen; "gid" kolonnen angives som primær nøgle, det defineres at der arbejdes med 2-dimensionalle koordinater og at tabellen skal modtage punktdata. Endelig defineres koordinatsystmet til SRID: 25832 (UTM 32N ETRF89

Herefter tilføjes et GiST til tabellen, se Kodebid 32.

```
CREATE INDEX temp_fortidsminder_the_geom_gist
  ON temp_fortidsminder
  USING gist
  (the_geom);
```

Kodebid 32: SQL kode til oprettelse af GiST indeks

De øvrige midlertidige tabeller er oprette efter samme skabelon.

6.4 Web- og Geoserver set-up

Udviklingen af en web-applikation kræver i sagens natur, at applikationen kan tilgås online. Derudover kræves det, at projektapplikationens Google API-funktioner har adgang til Googles servere via internettet. Online adgang sikres gennem en web-server, som enten kan hostes hos en kommercie-ludbyder, eller som det er tilfældet i dette projekt, ved at en webserver køres på en maskine som kan tilgås via internettet.

Nedenstående figur illustrerer projektets server set-up, se Figur 31. En lokal desktop står for afviklingen af Web-, Geo- og DataBaseServeren, og tilgås af eksterne klienters browsere via internettet.

Figur 31: Server set-up: En lokal desktop, som tilgås via internettet, agerer server for afvikling af både web-applikation, database og kortserver

6.4.1 Webserver

Til afvikling af selve web-applikationen benyttes Open Source krydsplatforms webserver-pakkeløsningen XAMPP, der er et akronym for:

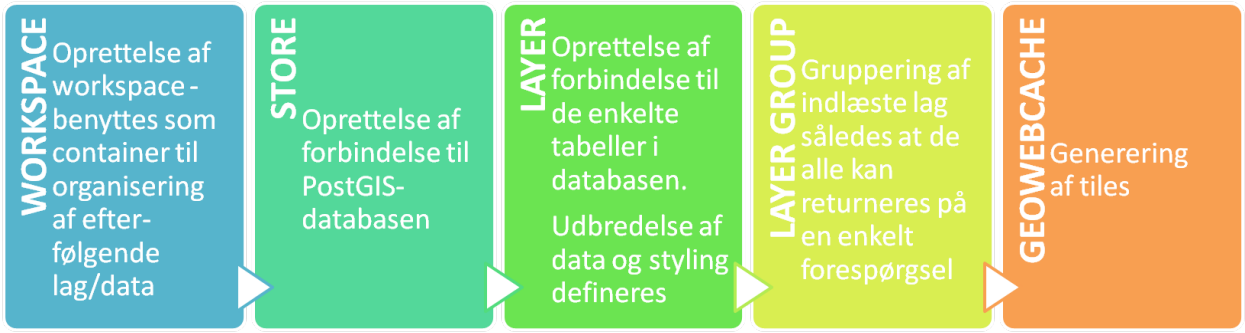
- X (læses som "kryds" => krydsplatform)
- Apache HTTP Server
- MySQL
- PHP
- Perl

Til afviklingen af projektapplikationen benyttes XAMPPs Apache server og PHP tolkningsprogram (eng. interpreter) – de øvrige af pakkeløsningens funktioner benyttes ikke.

Som tidligere beskrevet, se afsnit 6.3 *Database set-up*, foregår datalagring i et PostgreSQL/PostGIS set-up, som også afvikles på den lokale desktop.

6.4.2 Geoserver

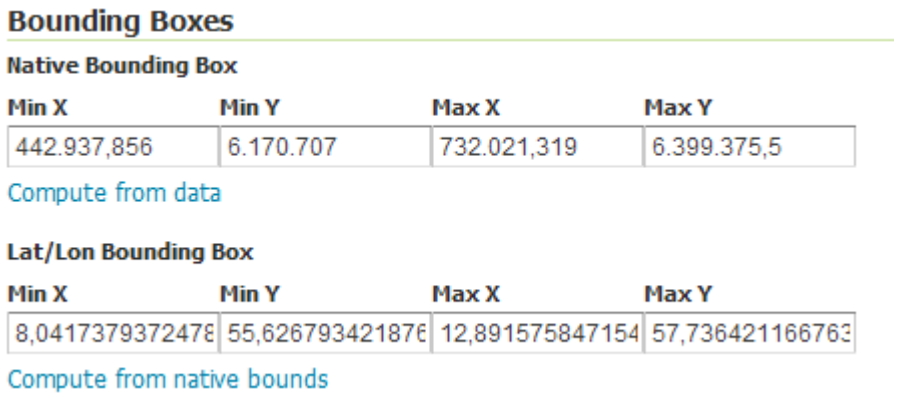
Til visualisering af PostGIS-databasen benyttes Open Source kortserveren GeoServer, som ligeledes afvikles på desktoppen. I det følgende vil kort blive gennemgået, hvordan håndtering og forberedelse til visualisering af projektdata sker via GeoServer. Nedenstående figur illustrerer de forskellige trin der er i opsætningen af GeoServer, se Figur 32. Kun de opsætningstrin hvor der er foretaget andet end standardopsætning, vil blive gennemgået i det følgende.



Figur 32: Opsætning af GeoServer trin for trin

Inden de enkelte tabeller kan visualiseres som lag via GeoServer, skal der bl.a. defineres parametre for udbredelse af data samt visualisering. Udbredelsen af data (eng. bounding box) er en af de ting som skal defineres når et lag oprettes.Under oftest vil man få GeoServer til at beregne bounding boxen udfra det givne datasæt. Dette er dog ikke en mulighed i dette projekt. Eksempelvis vil de midlertidige tabeller i udgangspunktet ikke have nogen udbredelse (da der ikke er noget data i dem), og det er derfor nødvendigt at definere udbredelse for tabellen, således at alt tilføjet data indenfor projektområdet vil være omfattet. Dette gør sig også gældende for de, tabeller som indeholder data – her skal det også sikres at evt. senere tilføjet data ikke falder uden for bounding box, og derved ikke vises for brugeren.

På denne baggrund defineres udbredelsen af alle projektets datalag som vist på Figur 33.



Figur 33: Bounding Box defineres i GeoServer

Inden et datalag kan oprettes og visualiseres skal det defineres hvordan data skal præsenteres for brugeren. Dette gøres ved at tilknytte en defineret style til datalaget. Til brug i projektet har GeoServers prædefinerede styles ikke været tilstrækkeligt til visualisering af det ønskede data. Det har derfor været nødvendigt, at definere nogle projektspecifikke styles.

Dette gøres i Georservers interface under "styles", hvor det ønskede udseende programmeres vha. SLD-kode (***S**tyled **L**ayer **D**escriptor*) som er et XML-baseret markup language. Følgende kodebid viser hvordan styling til datalaget *beskyt_linje* er defineret, se Kodebid 33 på næste side.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0"
    xsi:schemaLocation="http://www.opengis.net/sld
StyledLayerDescriptor.xsd"
    xmlns="http://www.opengis.net/sld"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <NamedLayer>
    <Name>Attribute-based line</Name>
    <UserStyle>
      <Title>SLD Cook Book: Attribute-based line</Title>
      <FeatureTypeStyle>
        <Rule>
          <Name>line</Name>
          <Title>Beskyttede vandløb</Title>
          <ogc:Filter>
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName>objekttkst</ogc:PropertyName>
              <ogc:Literal>Beskyttede vandløb</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <LineSymbolizer>
            <Stroke>
              <CssParameter name="stroke">#33CCFF</CssParameter>
              <CssParameter name="stroke-width">1</CssParameter>
            </Stroke>
          </LineSymbolizer>
          <PointSymbolizer>
               <Graphic>
                  <Mark>
                     <Fill>
                        <CssParameter name="fill-opacity">0</
CssParameter>
                     </Fill>
                  </Mark>
               </Graphic>
            </PointSymbolizer>
        </Rule>
      </FeatureTypeStyle>
      --- Kode til styling af beskyttede sten- og jorddiger udeladt
---    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

Kodebid 33: Styling af datalaget *beskyt_linje*

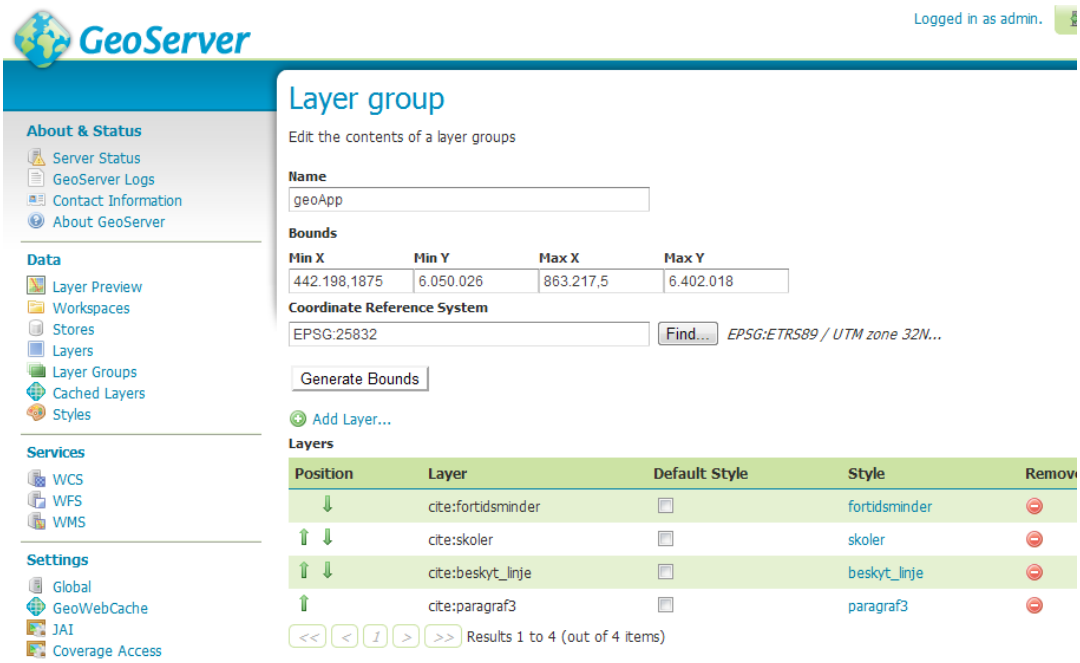
Ovenstående kodeeksempel viser en betinget styling af datalaget *beskyt_linje*. Vha. tagget `<Rule>` oprettes en betingelse for hvordan de enkelte linjer skal styles. Herefter defineres det, at laget skal styles på baggrund af tabelkolonnen *objekttkst* og at den pågældende regel, gælder for de tilfælde hvor indholdet i *objekttkstk* er *Beskyttede vandløb* se Kodebid 34.

```
<ogc:Filter>
  <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>objekttkst</ogc:PropertyName>
    <ogc:Literal>Beskyttede vandløb</ogc:Literal>
  </ogc:PropertyIsEqualTo>
</ogc:Filter>
```

Kodebid 34: Objekttkst - Beskyttede vandløb

Efterfølgende defineres visualiseringen under tagget `<LineSymbolizer>` med gængse CSS-parametre. På samme måde er der udarbejdet styling til projektets øvrige datalag.

For at minimere antallet af forespørgsler til GeoServeren – og dermed maksimere hastigheden – gruppes lagene *beskyt_linje*, *fortidsminder* og *skoler*. Dette gøres let vha. *Layer Groups* i GeoServer. Herunder vises den valgte opsætning, se Figur 34.



Figur 34: GeoServer: opsætning af *Layer Group*

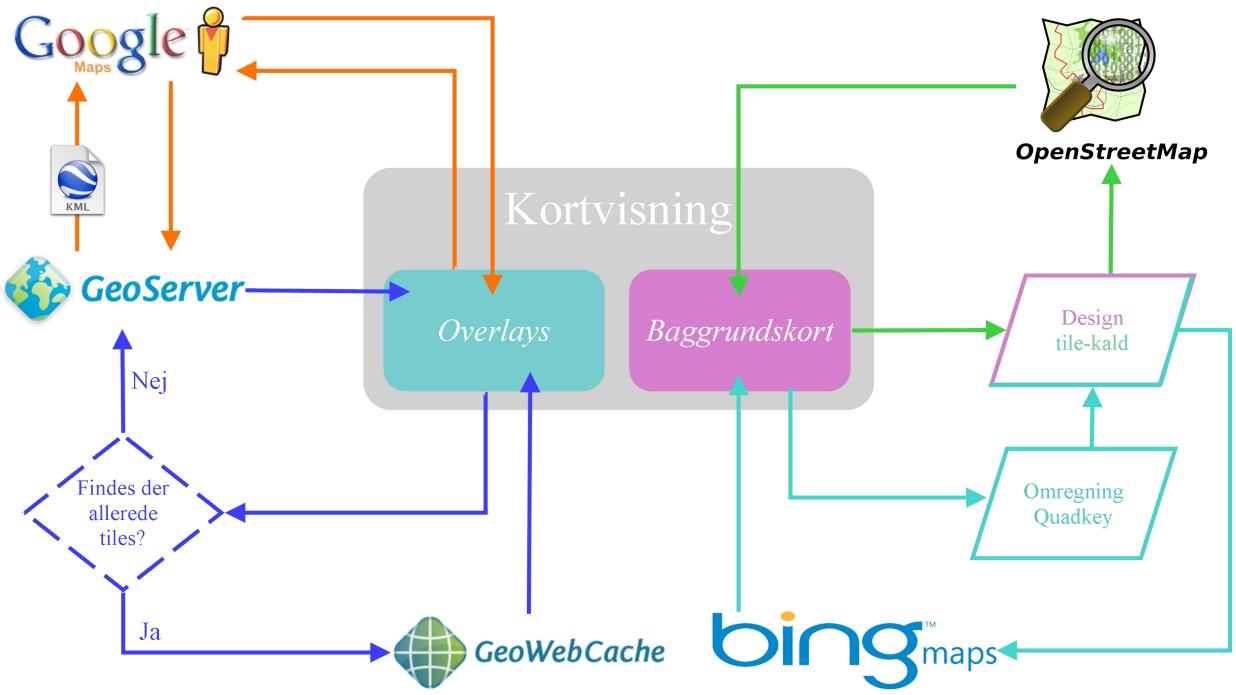
I dette tilfælde kan GeoServer bruges til automatisk at danne bounding boksen, da det ved oprettelsen af de enkelte lag er sikret, at data i hele projektområdet er indbefattet. Bemærk desuden at rækkefølgen som de enkelte lag stå i listen, vil være den som der dannes et billede ud fra, når der spørges til GeoServer. For at repræsentere data bedst muligt, er det valgt at have flader i bunden, herefter linjer og punkterne øverst.

For at mindske svartiden på en forespørgsel yderligere, benyttes GeoWebCache som er en integreret del af GeoServer. GeoWebCache lagrer genererede tiles, hvilet bevirker at GeoServer ikke skal generere et billede på ny hver gang det forespørges. I stedet vil systemet når, det får en forespørgsel, først tjekke om der allerede ligger et billede klar inden forespørgslen evt. sendes videre til bearbejdelse af GeoServeren. Denne proces sikre en langt hurtige svartid til applikationen.

Ulempen er, som beskrevet tidligere, at der ved opdatering/tilføjelse af data, skal genereres nye tiles for de berørte områder, hvilket er en tidskrævende proces. Via GeoWebCaches genereres der tiles af den førnævnte layer group, således at der ligger billeder klar, lige meget hvor der forespørges i projektområdet. Derved sikres de hurtigst mulige svartider til projektapplikationen.

6.5 Kortimplementering

I det følgende beskrives, hvorledes projektapplikationens kortvisning funger. Her vil bl.a. blive beskrevet hvordan Google Maps API v.3 integreres i Sencha Touch frameworket, tilføjelse af baggrundskort fra forskellige udbydere, samt hvordan overlays i form af både WMS og KML bliver integreret. Nedenstående figur viser hvorledes kortvisningsdelen af applikationen er opbygget, se Figur 35.



Figur 35: Opsætning af baggrundskort og overlays

6.5.1 Opsætning af basiskort med WMS og KML overlays

Google Maps API v.3 er en standardkomponent i Sencha Touch. Der er derfor utrolig enkelt at få et simpelt Google-kort vist i Sencha Touch. Nedenstående kode viser, hvorledes opsætningen af basiskortet i projektet er kodet, se Kodebid 35

```
var mapdemo = new Ext.Map({    //
      title: 'map',
      useCurrentLocation: false,
      mapOptions : {
            center : new google.maps.LatLng(positionArr[0],
positionArr[1]),
            minZoom: 1,
            zoom : positionArr[2],
            mapTypeId : positionArr[3],
            mapTypeControl: false,
            navigationControl: (Ext.is.Andriod? true : false),
            },
```

Kodebid 35: Opsætningen af basiskortet

Som det ses af ovenstående kode, er opsætningen af basiskortet i Sencha Touch, stort set identisk med opsætningen af et helt almindelig Google-kort. Dog skal det bemærkes at kortindstillingerne defineres inden i Sencha Touch komponenten `Ext.Map`, ligesom title og kontrollen (`Ext.is.Andriod?`) af om applikationen benyttes på en enhed med Android-styresystem også er Sencha Touch relaterede. Kontrollen i forhold til Android skyldes, at disse enheder kan have problemer i forhold til zoom-funktionaliteten ved brug af *pinch-to-zoom*. Derfor tilføjes to zoom-knapper i brugerfladen, hvis den benyttede enhed benytter styresystemet Android. Det ses desuden, at variablerne `center`, `zoom`, og `mapTypeId` får deres respektive værdier fra andre variable (`positionArr[]`), defineret et andet sted i koden. Dette set-up gennemgås senere.

Basiskortet tilføjes desuden en række *listeners* – funktioner som "lytter" på kortet, og køres når en given betingelse opfyldes, se Kodebid 36.

```
listeners : {
      maprender: function(extmap, map){
            ExtMap.MapTypes.registerMaps(map);

            google.maps.event.addListener(map, 'maptypeid_changed',
function() {
                  ExtMap.Storage.saveLastPosition(map)
            });

            google.maps.event.addListener(map, 'center_changed',
function() {
                  ExtMap.Storage.saveLastPosition(map)
            });

            google.maps.event.addListener(map, 'zoom_changed',
function() {
                  ExtMap.Storage.saveLastPosition(map)
});
```

Kodebid 36: Kortændringslisteners

Ovenstående kode viser, at funktionen `ExtMap.Storage.saveLastPosition(map)` køres hvis enten *korttypen*, *kortcenter* eller *zoomniveau* ændres. Denne funktion bruges til at gemme parametrene for den nuværende kortvisning, således at man ved en opdatering af kortet, stadig vil bibeholde den samme visning.

Funktion gemmer data i en enhedens lokale hukommelse. For at dette kan lade sig gøre, skal der først defineres en model for det lagrede data. Dette gøres som vist i nedenstående kode, se Kodebid 37

```
Ext.regModel('extmapLastPosition', {
    fields: [{
        name: 'id',
        type: 'int'
    }, {
        name: 'lat'
    }, {
        name: 'lng'
    }, {
        name: 'zoom',
        type: 'int'
    }, {
        name: 'maptype',
        type: 'string'
    }]
});
```

Kodebid 37: Model til lagring af data i enhedens lokale hukommelse

Efterfølgende defineres forbindelsen til den lokale hukommelse, hvor der oprettes en lokal data store på baggrund af ovenstående model. Dette gøres som vist i Kodebid 38.

```
ExtMap.Storage.positionStore = new Ext.data.Store({
    model: 'extmapLastPosition',
    proxy: {
        type: 'localstorage',
        id: 'positionStorage',
        proxy: {
            idProperty: 'id'
        }
    }
      });
```

Kodebid 38: Forbindelse til den lokale hukommelse

Herunder ses hvordan førnævnte funktion, interagerer med den definerede data store, se Kodebid 39

```
ExtMap.Storage.saveLastPosition = function (map) {
    ExtMap.Storage.positionStore.getProxy().clear();
    var newPosition = Ext.ModelMgr.create({
        id: 1,
        lat: map.getCenter().lat(),
        lng: map.getCenter().lng(),
        zoom: map.getZoom(),
        maptype: map.getMapTypeId()
    }, 'extmapLastPosition');
    ExtMap.Storage.positionStore.add(newPosition);
    ExtMap.Storage.positionStore.sync()
};
```

Kodebid 39: Interaktion mellem funktion og data store

Ovenstående funktion sletter først hvad der måtte være af gamle registreringer i data storen. Herefter indhentes information om kortets aktuelle korttype, kortcenter og zoomniveau. Disse informationer tilføjes data storen som en ny position, hvorefter data storen synkroniseres. Hermed indeholder data storen kun information om den aktuelle kortvisning. Funktionen til lagring af information om den aktuelle kortvisning er lavet med inspiration og hjælp fra Alper Dinçer, der bl.a. står bag http://www.extmap.com/. Med ovenstående set-up på plads, er det nu tid til at kigge nærmere på hvordan basiskortet tilføjes overlays af data fra databasen.

KML-overlay bruges til at vise data fra de midlertidige databasetabeller. Denne løsning benyttes som sagt, fordi WMS-laget forspørges i prædefinerede tiles, hvorfor opdatering er tids- og ressourcekrævende.

Det er med Google Maps API'en muligt at indlæse et KML-lag ovenpå basiskortet. Det sker via følgende kode, se Kodebid 40.

```
var KMLLayer = new google.maps.KmlLayer( '.../KMLpoints.kml',
{preserveViewport:true});
KMLLayer.setMap(mapdemo.map);
```

Kodebid 40: KML overlay

For at ovenstående funktion kan afvikles, kræves det at Google kan få adgang til den specificerede KML-fil, desuden har Google en indbygget begrænsning mht. til KML-filens størrelse på 10 MB, hvilket dog ikke skønnes at være et problem i forhold til set-up'et i projektapplikationen. Ovenstående kode specificerer desuden, at den aktuelle kortvisning skal bibeholdes.

WMS-overlay bruges i projektapplikationen til visualisering af databasegrundtabellerne; *paragraf3*, *fortidsminder* og *bestyk_linje* som er samlet til en enkelt *layer-group*, som beskrevet i afsnit 6.4 *Web- og Geosesrver set-up*. Koden på næste side viser, hvorledes projektapplikationen forespørger tiles gennem GeoWebCache og GeoServer, se Kodebid 41.

```
var imageMapTypeOptions = {
      getTileUrl: function(coord, zoom) {
            return 'http://2.107.248.183:8080/geoserver/gwc/service/
gmaps?layers=cite:paragraf3&zoom=' + zoom + '&x=' + coord.x + '&y='
+ coord.y + '&format=image/png';
            },
            tileSize: new google.maps.Size(256, 256),
            isPng: true,
            opacity: 0.5
            }

var tiledImageMap = new google.maps.ImageMapType(imageMapTypeOption
s);

map.overlayMapTypes.insertAt(0, tiledImageMap);
            }
      }
});
```

Kodebid 41: WMS overlay

`getTileUrl` er ligeledes en funktion i Google Maps API'en. Kortvisningens aktuelle zoomniveau og koordinater indsættes i forespørgslen til GeoWebCache/GeoServer. Herefter defineres tilestørrelse til 256x256 pixel, hvilket er standard for Google Maps. Dernæst defineres billedformatet og gennemsigtighed for overlayet. Slutteligt defineres WMS-laget til at ligge på første position over basiskortet.

6.5.2 Baggrundskort

For at tilføre projektapplikationen ekstra funktionalitet og brugbarhed, vælges det at give brugeren mulighed for at vælge mellem forskellige baggrundskort. Dette gøres, fordi der er stor forskel på indholdet af de forskellige kort, og ikke mindst fordi, at der på flyfotos ofte er være ganske stor forskel på kvaliteten i forskellige områder og zoomniveauer. I projektet vælges der derfor, at tilføje baggrundskort til den eksisterende Google-løsning fra udbyderene *Bing* og ***O**pen**S**treet**M**ap*(OSM). Derudover tilføjes et blankt baggrundskort, som vil fremme visningen af overlay-data. Alle baggrundskort vil stadig køre med Googles "motor", men selve kortet vil variere. Koden til styring af baggrundskort er ligeledes lavet med inspiration og hjælp fra Alper Dinçer.

6.5.3 Bing maps

Bing maps benytter en QuadKey-struktur til lagring af tiles til deres kortservices. Der skal derfor foretages nogle beregninger, inden tiles fra Bing kan bruges sammen med Google Maps API'en. En uddybende forklaring af Bings tilestruktur og uddybning af efterfølgende beregninger, kan findes på http://msdn.microsoft.com/en-us/library/bb259689.aspx. Her vil kun den overordnede funktionalitet blive beskrevet.

Nedestående kode bruges til omregning fra Tile-koordinater til Quadkey, se Kodebid 42.

```
ExtMap.Utils.TileToQuadKey = function (x, y, zoom) {
    var quad = "";
    for (var i = zoom; i > 0; i--) {
        var mask = 1 << (i - 1);
        var cell = 0;
        if ((x & mask) != 0) cell++;
        if ((y & mask) != 0) cell += 2;
        quad += cell
    }
    return quad
};
```

Kodebid 42: Quadkey omregning

Herefter konstrueres det kald, der bruges til en forespørgsel hos Bing, se Kodebid 43. Bemærk at funktionen defineret i Kodebid 42 bruges i denne beregning og får koordinater og zoomniveau som input.

```
ExtMap.MapTypes.getMapTilesQuad = function (coord, zoom, ownerDocument, url1, url2,
url3) {
    var div = ownerDocument.createElement('DIV');
    var tempQuad = ExtMap.Utils.TileToQuadKey(coord.x, coord.y, zoom);
    var quadTile = '000000';
    quadTile += (parseInt(coord.y.toString(2) * 2) + parseInt(coord.x.toString(2)));
    quadTile = quadTile.substring(quadTile.length - zoom, quadTile.length);
    tileUrl = url1 + quadTile.substring(quadTile.length - 1, quadTile.length) + url2
+ tempQuad + url3;
    div.innerHTML = '<img src="' + tileUrl + '">';
    return div
};
```

Kodebid 43: Forespørgsel hos Bing

Endeligt laves den egentlige forespørgsel til Bing. Nedenstående kode viser hvordan Bings road map kaldes, se Kodebid 44.

```
ExtMap.MapTypes.bingMapsRoadLayer = function () {};
ExtMap.MapTypes.bingMapsRoadLayer.prototype.tileSize = new google.
maps.Size(256, 256);
ExtMap.MapTypes.bingMapsRoadLayer.prototype.maxZoom = 19;
ExtMap.MapTypes.bingMapsRoadLayer.prototype.getTile = function
(coord, zoom, ownerDocument) {
    return ExtMap.MapTypes.getMapTilesQuad(coord, zoom,
ownerDocument, 'http://ecn.t', '.tiles.virtualearth.net/tiles/r',
'.png?g=563&mkt=en-us&lbl=l1&stl=h&shading=hill&n=z')
      };
ExtMap.MapTypes.bingMapsRoadLayer.prototype.name = "Bing Maps";
ExtMap.MapTypes.bingMapsRoadLayer.prototype.alt = "Bing Maps Road";
```

Kodebid 44: Kald til Bing Road Map

På samme må som ovenstående, kaldes hhv. Bings luftfoto og - hybridkort.

6.5.4 Openstreetmap

Kaldet til OpenStreetMap, se Kodebid 45 er noget mere ligetil – og minder meget om WMS-kaldet til Geoserver, se Kodebid 41.

```
ExtMap.MapTypes.getMapTilesOSM = function (coord, zoom,
ownerDocument) {
    var div = ownerDocument.createElement('DIV');
    var servers = new Array("a", "a", "b", "c");
    var rand_no = Math.ceil(Math.random() * 3);
    tileUrl = 'http://' + servers[rand_no] + '.tile.openstreetmap.
org/' + zoom + '/' + coord.x + '/' + coord.y + '.png';
    div.innerHTML = '<img src="' + tileUrl + '">';
    return div
};
```

Kodebid 45: Kald til OSM

Som det ses af ovenstående kode, er URL-kaldet sammensat af en tilfældigt valgt server (a, b, eller c plus et tilfældigt tal). Desuden kræver kaldet input i form af kortets aktuelle centerkoordinater og zoomniveau. Herefter kaldes OpenStreetMap på samme vis som beskrevet i Kodebid 44

6.5.5 Blankt baggrundskort

For at give brugeren muligheden for kun at se data fra databasen, uden forvirring fra et egentlig baggrundskort, laves et set-up som genererer neutralt udseende tiles. Følgende kode viser hvordan det er gjort, se Kodebid 46

```
ExtMap.MapTypes.blankMapType = function () {};
ExtMap.MapTypes.blankMapType.prototype.tileSize = new google.maps.
Size(256, 256);
ExtMap.MapTypes.blankMapType.prototype.maxZoom = 19;
ExtMap.MapTypes.blankMapType.prototype.getTile = function (coord,
zoom, ownerDocument) {
    var div = ownerDocument.createElement('DIV');
    div.style.width = this.tileSize.width + 'px';
    div.style.height = this.tileSize.height + 'px';
    div.style.borderColor = '#AAAAAA';
    return div
};
ExtMap.MapTypes.blankMapType.prototype.name = "Blank Map";
ExtMap.MapTypes.blankMapType.prototype.alt = "Blank Map Type";
```

Kodebid 46: Blanke tiles

Genereringen af blanke tiles foregår stort set på samme måde som beskrevet i Kodebid 44. Dog kommer tiles ikke fra en web-kilde, men defineres i stedet sidst i koden. Det ses af ovenstående af der oprettes tiles med en størrelse på 256 x 256 pixels med baggrundsfarven (#AAAAAA – lys grå).

6.5.6 Kald af baggrundskort

Valg af baggrundskort forgår via listevisningen i menupunktet "Choose map". Når et baggrundskort vælges i listen, eksekveres følgende script; se Kodebid 47

```
ExtMap.UI.mapTypeList.on('itemtap', function (list, index, item, e)
      {
            mapPanel.setActiveItem(new Ext.Panel({
            items: [mapdemo]
      }), {
            type: 'fade',
            duration: 1000
});
mapdemo.map.setMapTypeId(list.store.getAt(index).get('typex'));
ExtMap.UI.mapListOverlay.hide()
});
```

Kodebid 47: Valg af baggrundskort fra liste

Ovenstående kode viser, at der ved valg af et nyt baggrundskort, indlæses et nyt kort med den valgte baggrundskorttype (*typex*). Skiftet til den nye kortvisning sker med en fade-funktion, som vare ét sekund. Den tidligere nævnte funktionalitet til lagring af kortvisningens aktuelle parametre, sørger for at kortets centerposition og zoomniveau bibeholdes, og kun baggrundskortet ændres.

6.6 Lokaliseringsfunktionalitet

I applikationens undermenu *Locate* findes funktionerne *Current location* og *Find Address*. Der centrerer kortet ud fra hhv. aktuelle position vha. XPS og ud fra en geokodet brugerangivet adresse.

6.6.1 Current location

Nedenstående Kodebid 48 viser koden bag *Current location* funktionen.

```
if  (listnode == 1) {
                var geo = new Ext.util.GeoLocation({
                autoUpdate: false,
                allowHighAccuracy: false,
                timeout: 6000,
                listeners: {
                        locationupdate: function (geo) {
                                mapdemo.map.setCenter(new google.maps.LatLng(geo.
latitude, geo.longitude));
                                mapdemo.map.setZoom(15);

                                Marker = new google.maps.Marker({
                                        icon: 'img/marker_blue.png',
                                        animation: google.maps.Animation.BOUNCE,
                                        map: mapdemo.map,
                                        position: new google.maps.LatLng(geo.latitude,
geo.longitude),
                                });
                                markersArray.push(Marker);

                                var geolat = geo.latitude
                                var geolng = geo.longitude
                                var geoaccu = geo.accuracy

                                Ext.Msg.alert('Coordinates', 'lat.' + geolat.
toFixed(6)+'<br> lng. '+geolng.toFixed(6) + '<br> accuracy. ' + geoaccu + ' m',
function () {window.setTimeout(clearAll,5000)});

},}});
        geo.updateLocation()
};
```

Kodebid 48: Current location

Som det ses benyttes klassen `Ext.util.GeoLocation`, der er et krydsbrowserkald defineret af Sencha Touch, som benyttes til at hente lokalitetsbestemmelsesinformationer. Kaldet er baseret på

W3Cs "Geolocation API specification" (Sencha(n.d) & World Wide Web Consortium 3, (n.d.)).

I applikationen opsættes kaldets parameter til:

- `autoUpdate: false` – Sikre at kaldet ikke opdaterer mere end én gang.
- `allowHighAccuracy: false` – Efter forsøg med denne parameter sat til både true og false, måtte vi konkludere, at vi ikke fik bedre nøjagtighed med `HighAccuracy` på. Derfor er denne parameter slået fra, da den trækker kraftigt på batteriet. Det må dog forventes at denne funktionalitet i fremtiden vil fungere bedre, i takt med at geolocate specifikationen bliver mere udbredt i mobile browser.
- `Timeout: 6000` – Sætter en timeout for kaldet på 6 sekunder.

```
listeners: {
      locationupdate: function (geo) {
      mapdemo.map.setCenter(new google.maps.LatLng(geo.latitude,
geo.longitude));
      mapdemo.map.setZoom(15);
```

Kodebid 49: location listeners

Funktionen opsættes med en listener på hvorvidt `geo`-funktionen, returnerer et succesfuldt `locationupdate` svar. Sker dette returneres to variable, `geo.latitude`(længdegrad) og `geo.logitude`(bredegrad), se Kodebid 49. Som benyttes til `setCenter(new google.maps.LatLng(geo.latitude, geo.longitude))`, hvilket er et Google Maps API kald, til at centrerer kortet omkring de givne koordinater. Yderligere sættes zoomniveauet til 15 med kort API kaldet `setZoom(15)`.

Herefter sættes en hoppende markør på de nye koordinater, og markøren pushes til `markersArray`, se Kodebid 50. MarkersArray benyttes til at fjerne alle markers i applikationen, når diverse "*Clear*" og "*cancel*" funktioner køres

```
Marker = new google.maps.Marker({
       icon: 'img/marker_blue.png',
       animation: google.maps.Animation.BOUNCE,
       map: mapdemo.map,
       position: new google.maps.LatLng(geo.latitude, geo.longitude),
```

Kodebid 50: Markør

Koordinaterne og `geo.accuracy`(fra geolocate kaldet) gemmes herefter ud som variable, der kan indgå i en tekst string. Disse variable benyttes i en *pop-up alert* (`Ext.Msg.Alert(...)`), der indeholder information om koordinater og præcision, se Kodebid 51

```
var geolat = geo.latitude
var geolng = geo.longitude
var geoaccu = geo.accuracy

Ext.Msg.alert('Coordinates', 'lat.' + geolat.toFixed(6)+'<br> lng.
'+geolng.toFixed(6) + '<br> accuracy. ' + geoaccu + ' m', function
() {window.setTimeout(clearAll,5000)});
```

Kodebid 51: Info pop-up

Afslutningsvis kalder *Pop-up alert* funktionen også en anden funktion, der rydder `markersArray` efter 5 sekunder, via funktionen `clearAll`, som er defineret et andet sted i *index.js*.

6.6.2 Find location

Kodebid 52 viser koden bag *Find location* funktionen.

```
ExtMap.Map.geocoder = new google.maps.Geocoder();


ExtMap.Utils.findAddress = function (map) {
      Ext.Msg.prompt("Find Address", "Please enter an address",
function (value, txtvalue) {
      ExtMap.Map.geocoder.geocode({
                  'address': txtvalue
            }, function (results, status) {
                  if (status == google.maps.GeocoderStatus.OK) {
                        mapdemo.map.setCenter(results[0].geometry.
location);
                        ExtMap.Map.addressMarker = new google.maps.
Marker({
                              icon: 'img/marker_blue.png',
                              animation: google.maps.Animation.
BOUNCE,
                              map: mapdemo.map,
                              position: results[0].geometry.location
                        });
                        mapdemo.map.setZoom(15);
                        console.log(results[0].geometry.location);
                        setTimeout(function () {
                              ExtMap.Map.addressMarker.setMap(null)
                         }, 5000)
                  } else {
                        Ext.Msg.alert('Alert', 'Geocoder failed due
to: ' + status, null)
                  }
            })
      },
      this,
      false,
      null,
      {maxlength: 100, autocapitalize : false, placeholder :
"Address, town, country..."}
      )
};
ExtMap.Utils.findAddress(mapdemo.map)
```

Kodebid 52: Find location

Inden selve funktionen opsættes defineres en ny global variabel, til at kalde på Google Maps Geocoder, se Kodebid 53.

```
ExtMap.Map.geocoder = new google.maps.Geocoder();
```

Kodebid 53: Google API geocoder

Dette er nødvendigt, for at kunne kalde geocoderen ind i funktionen senere. At geokode er en proces, til at konvertere adresser(fx "Strandvejen 12, Hellerup, DK") til geografiske koordinater(længde- og breddegrader).

```
Ext.Msg.prompt("Find Address", "Please enter an address", function
(value, txtvalue) {
      ExtMap.Map.geocoder.geocode({
                  'address': txtvalue
```

Kodebid 54: Promt

Når brugeren trykker *find address*, kommer en prompt om at indtaste en adresse. Kodebid 54 viser hvordan denne prompt er opbygget, med en overskrift(*"Find Address"*), en placeholder tekst(*"Please enter an address"*) og en geocoder funktion der tager to inputs(`value` og `txtvalue`).

`Txtvalue` er den adresse, som brugeren netop har indtastet, og denne sendes nu til Google Maps Geocoder via den førnævnte globale variabel (`ExtMap.Map.geocoder.geocode`). Herfra er *Find adresse* funktionen opbygget efter "`if/else`" princippet. se Kodebid 55.

```
if (status == google.maps.GeocoderStatus.OK) {
      mapdemo.map.setCenter(results[0].geometry.location);
      ExtMap.Map.addressMarker = new google.maps.Marker({
            icon: 'img/marker_blue.png',
            animation: google.maps.Animation.BOUNCE,
            map: mapdemo.map,
            position: results[0].geometry.location
      });
      mapdemo.map.setZoom(15);
      console.log(results[0].geometry.location);
      setTimeout(function () {
             ExtMap.Map.addressMarker.setMap(null)
       }, 5000)
```

Kodebid 55: If delen

Kodebid 55 viser hvad der sker, når geokoderen returnerer et succesfuld resultat (`status == google.maps.GeocoderStatus.OK`). Som det ses centreres kortet om de nye koordinater (`mapdemo.map.setCenter(results[0].geometry.location);`) og en hoppende markør placeres, efter samme princip, som beskrevet for *Current location* funktionen.

```
else {
      Ext.Msg.alert('Alert', 'Geocoder failed due to: ' + status,
null)
}
```

Kodebid 56: Else delen

Kan geokoderen ikke finde et resultat, køres der en *pop-op alert*, med en overskrift (`'Alert'`) samt en forklaring på hvorfor geokoderen fejlede (`'Geocoder failed due to: ' + status'`).

6.6.3 Get Coordinates

I applikationen findes der en tredje funktionalitet, der også er relateret til lokalisering, *"Get coordinates"*, der findes under *Tools*. Denne fungerer ved, at brugeren klikker på kortet, hvorefter en *pop-op alert* informerer brugeren om stedets koordinater. Kodebid 57 viser hvordan.

```
google.maps.event.addListenerOnce(mapdemo.map, "mousedown", function(evt) {
      markerLat = evt.latLng.lat();
      markerLng = evt.latLng.lng();
      Ext.Msg.alert('Coordinates', 'lat: ' + markerLat.toFixed(6) +'<br>
lng: '+markerLng.toFixed(6), Ext.emptyFn);
            });
      }
      ExtMap.UI.toolListOverlay.hide()
});
```

Kodebid 57: Get Coordinates

Når *Get Coordinates* værktøjet vælges aktiveres en engangslistner til kortet, der registrerer *mousedown* events. Ved en sådan event gemmes først bredde- og længdegraderne i variable (`markerLat = evt.latLng.lat()` og `markerLng = evt.latLng.lng()`), der derefter skrives ud vha. en *pop-op alert* besked (`Ext.Msg.alert('overskrift', 'brødtekst', funktion);`).

6.7 Målefunktioner

I applikationen findes der to værktøjer til opmåling af længde og areal (*Measure length* og *Measure area*), der er bygget ud fra de principper, Jason Sanford beskriver på sin hjemmeside (SANFORD 2011). Funktionerne er grundlæggende ens, den eneste forskel er, at linje funktionen ikke har fyld, og at arealet ikke bliver vist i infobaren i toppen af skærmen, se Figur 36.

Figur 36: Find area 1 (tv) og Find line 2 (th).

Measure Area og *Measure Line* gør brug af Google API'ens *Geometry* bibliotek. Der kan tilføjes til Google Map kaldet ved at tilføje `&libraries=geometry`, se Kodebid 58.

```
src="http://maps.google.com/maps/api/js?sensor=true&libraries=geometry"
```

Kodebid 58: Tilføj geometry biblioteket

For at kunne holde styr på de oprettede linjer, polygoner og markører, oprettes objektet `measure`, der indeholder et *MVCarray* for hver af de rumlige typer, se Kodebid 59.

```
var measure = {
      mvcLine: new google.maps.MVCArray(),
      mvcPolygon: new google.maps.MVCArray(),
      mvcMarkers: new google.maps.MVCArray(),
      line: null,
      polygon: null
};
```

Kodebid 59: Measure objekt

Herefter køres measure funktionen measureAdd se Kodebid 60.

```
function measureAdd(latLng) {
      var marker = new google.maps.Marker({
            map: mapdemo.map,
            position: latLng,
            draggable: true,
            raiseOnDrag: false,
            title: "Drag me to change shape",
            icon: new google.maps.MarkerImage("images/measure-
vertex.png", new google.maps.Size(9, 9), new google.maps.Point(0,
0), new google.maps.Point(5, 5))
});
```

Kodebid 60: MeasureAdd

Først tilføjes en flytbar markør til baggrundskortet `mapdemo.map`. Herefter pushes koordinaterne (`latLng`) fra de tilføjede markører til de førnævnte arrays, se Kodebid 61.

```
measure.mvcLine.push(latLng);
measure.mvcPolygon.push(latLng);
measure.mvcMarkers.push(marker);
```

Kodebid 61: mvcArrays

Nu oprettes et indeks med de koordinater vi netop har pushes ind i array'ene, hvorved vi kan opdatere arrayene, hvis brugeren senere skulle flytte på markørerne, se Kodebid 62.

```
var latLngIndex = measure.mvcLine.getLength() - 1;
```

Kodebid 62: Measure objekt

Når brugeren flytter på et punkt/markør, kører en listener, ses i Kodebid 63. Listeneren finder først det givne punkt vha. indekset (`latLngIndex`) og opdaterer det herefter med de nye koordinater (`evt.latLng`). Endelig køres `measureCalc` (beskrives neden for).

```
google.maps.event.addListener(marker, "drag", function(evt) {
      measure.mvcLine.setAt(latLngIndex, evt.latLng);
      measure.mvcPolygon.setAt(latLngIndex, evt.latLng);
      });

google.maps.event.addListener(marker, "dragend", function() {
      if (measure.mvcLine.getLength() > 1) {
      measureCalc();
      }
      });
```

Kodebid 63: Drag listeners

Herefter defineres hvornår og hvordan linjer og polygoner tegnes på kortet, se Kodebid 64.

```
if (!measure.line) {
            measure.line = new google.maps.Polyline({
                  map: mapdemo.map,
                  clickable: false,
                  strokeColor: "#6069EB",
                  strokeOpacity: 1,
                  strokeWeight: 3,
                  path:measure. mvcLine
            });
      }
if (measure.mvcPolygon.getLength() > 2) {
            if (!measure.polygon) {
                  measure.polygon = new google.maps.Polygon({
                        clickable: false,
                        map: mapdemo.map,
                        fillColor: "#6069EB",
                        fillOpacity: opacity,
                        strokeOpacity: 0,
                        paths: measure.mvcPolygon
                  });
            }
      }
}
if (measure.mvcLine.getLength() > 1) {
      measureCalc();
}
}
```

Kodebid 64: Hvordan linjer optegnes

Ovenstående kode angiver, at hvis linje array'et(`mvcLine`) har en længde på over én og der ikke allerede er en linje, tegnes der en ny linje(`new google.maps.Polyline`) på kortet. Ligeledes angives det, at hvis polygon array'et(`mvcPolygon`) har en længde på over to, og der ikke allerede findes en linje, tegnes der et nyt polygon(`new google.maps.Polygon`) på kortet. Endelig an-

giver funktionen også, at hvis `mvcLine` har en længde på over en, skal funktionen `measureCalc` køres.

`MeasureCalc` funktionen er den der laver den faktiske længde- og arealberegning, med hjælp fra Google Maps Geometry biblioteket, se Kodebid 65.

```
function measureCalc() {
      length = google.maps.geometry.spherical.computeLength(measure.
line.getPath());
      jQuery("#span-length").text(length.toFixed(1));
      jQuery("#span-length2").text(length.toFixed(1));

      if (measure.mvcPolygon.getLength() > 2) {
            area = google.maps.geometry.spherical.
computeArea(measure.polygon.getPath());
            jQuery("#span-area").text(area.toFixed(1));
      }
}
```

Kodebid 65: Measure objekt

Som det ses håndteres al beregningen af geometry biblioteket, og via jQuery gemmes resultatet ud, således at det kan benyttes i infobaren.

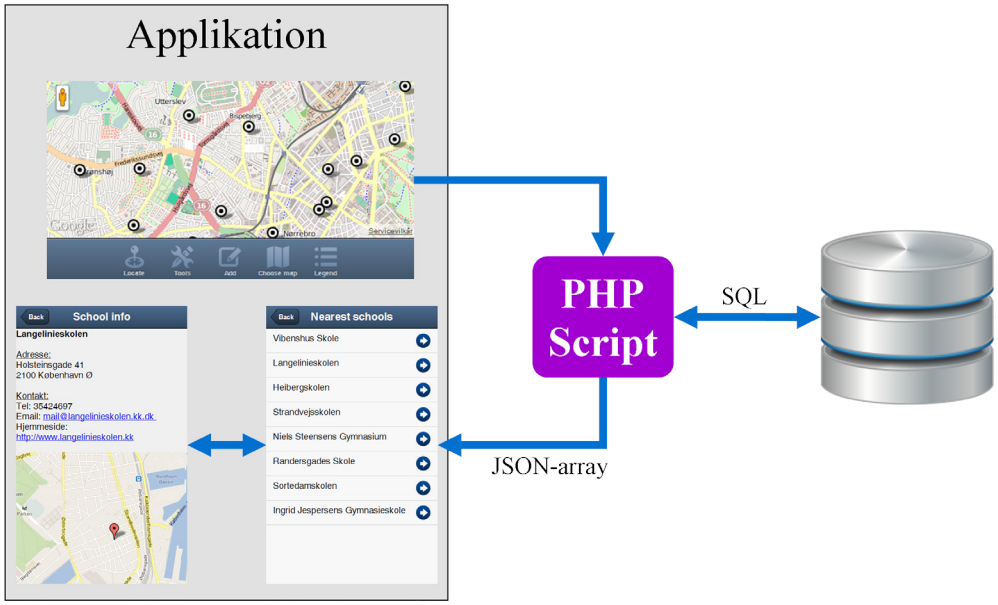
6.8 Rummelige forespørgsler

Ved hjælp af projektapplikationen er det muligt for at foretage tre simple rummelige analyser:

- *Near Point* – returnerer information om et givent datalag indenfor en brugerdefineret afstand af et af brugeren valgt punkt.
- *Near Line* – som ovenstående. Brugeren definerer en linje i stedet for et punkt.
- *Within Polygon* – returnerer information om data indenfor et af brugeren defineret polygon.

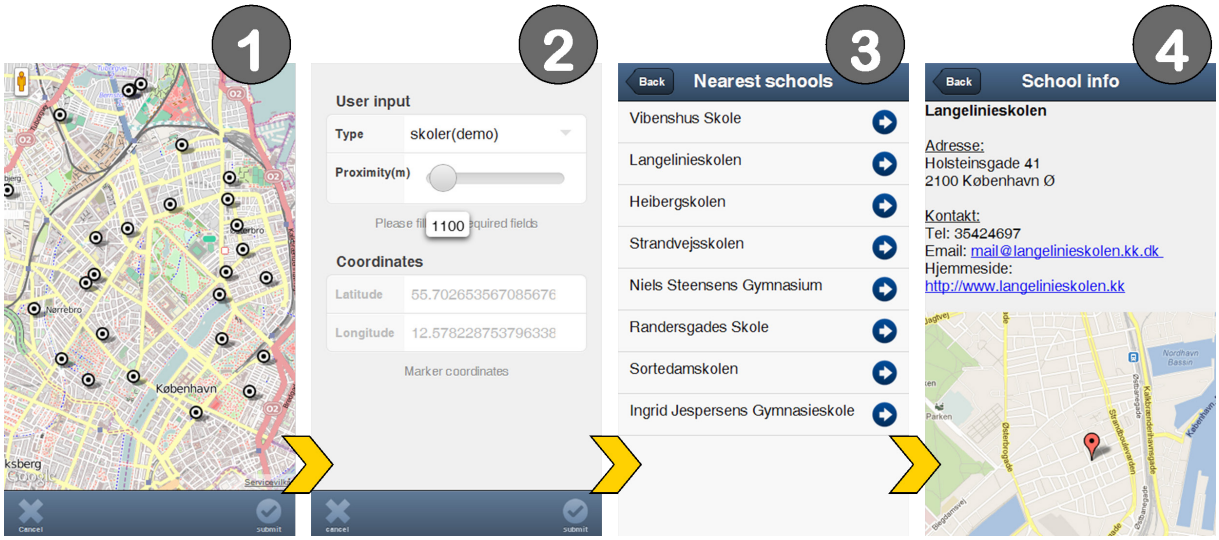
Koden til de tre ovenstående funktioner er langt hen af vejen det samme, så kun funktionen *Near Point* vil blive beskrevet yderligere i det følgende.

De rummelige forespørgsler bygger videre på funktionaliteten fra målefunktionerne beskrevet i foregående afsnit 6.7 *Målefunktioner*. Disse funktioner vil derfor ikke blive berørt yderligere, ligesom koden for design af selve brugergrænsefladen ikke vil blive gennemgået, men blot illustreret med screen shots. Fokus vil altså være, på det der sker fra de variable "forlader" applikationen til der returneres et resultat. Nedenstående figur illustrerer denne proces, se Figur 37.



Figur 37: Serverprocessering af rummelig forespørgsel

Inden der gås i dybden med den egentlige kode, kommer der her, for forståelsens skyld, en kort gennemgang af hvorledes en rummelig analyse laves og præsenteres for brugeren, se Figur 38.



Figur 38: Trinsvis illustration af skærmbilleder i forbindelse med en rummelig forespørgsel (her "Near Point")

Ovenstående figur viser de fire trin brugeren skal igennem, når der vælges at lave en rummelig analyse.

Her forklares processen ud fra analysen *Near Point*:

- Trin 1: Brugeren definerer det punkt, der skal danne udgangspunkt for analysen. Koordinater for det valgte punkt føres videre til næste skærm.
- Trin 2: Her definerer brugeren i hvilket data der ønskes søgt, samt en søgeradius fra det valgte punkt.
- Trin 3: Resultatet af analysen præsenteres for brugeren på listeform. Brugeren har mulighed for at klikke på de enkelte listeelementer for yderligere information.
- Trin 4: Her vises uddybende information fra det valgte listeelement.

Selve processeringen af den rummelige forespørgsel, se Figur 37, sker altså mellem trin 2 og trin 3. Der vil i det kommende blive set nærmere på, hvorledes den rummelige analyse foretages.

Første skridt er at indhente de brugerdefinerede variable. Dette sker ved hjælp af nedenstående handler, som eksekveres når brugeren trykker submit i Trin 2, se Kodebid 66

```
var submitProxHandler = function(button, event) {
      ProxType = submitProxForm.getValues().type,
      ProxRadius =  submitProxForm.getValues().radius,
      ProxLat = submitProxForm.getValues().lat,
      ProxLng = submitProxForm.getValues().lng,

      ProxUrl = '/29feb/php/serverRadius.php?lat='+ProxLat+'&lng='+P
roxLng+'&type='+ProxType+'&radius='+ProxRadius,

      searchStore.proxy.url = ProxUrl;
      searchStore.load();
      Panel.setActiveItem(proxListPanel, {type: 'slide', direction:
'left'})

...Kode undladt...
};
```

Kodebid 66: NearPoint handler

De ønskede parametre gemmes i variable, og sendes herefter videre til PHP-scriptet *serverRadius.php*(Bilag 8), se Kodebid 67.

```
<?php
      require("dbinfo.php");
       $conn_string = "host=$ip port=5432 dbname=$database
user=$username password=$password";
       $dbconn = pg_connect($conn_string);

      $search = array();

      $lat = $_GET['lat'];
      $lng = $_GET['lng'];
      $radius = $_GET['radius'];
      $type = $_GET['type'];

      if ($type == "skoler"){
      $post = pg_query($dbconn,
      "SELECT * FROM skoler WHERE ST_DWithin(ST_Transform(the_geom,
25832),ST_Transform(ST_GeomFromText('POINT($lng $lat)', 4326),
25832), $radius)");
      }
      else{
      $post = pg_query($dbconn,
      "SELECT * FROM fortidsminder WHERE ST_DWithin(ST_
Transform(the_geom, 25832),ST_Transform(ST_GeomFromText('POINT($lng
$lat)', 4326), 25832), $radius)");
      };

      while ($row =pg_fetch_object($post))  {
            $search[] = $row;
      };

echo json_encode(array('search' =>$search));
?>
```

Kodebid 67: serverRadius.php

PHP-scriptet opretter som det første forbindelse til databasen, se Kodebid 68, hvilket kræver at filen *dbinfo.php* (Bilag 4) er tilgængelig, da denne indeholder de nødvendige login-oplysninger.

```
require("dbinfo.php");
      $conn_string = "host=$ip port=5432 dbname=$database
user=$username password=$password";
      $dbconn = pg_connect($conn_string)
```

Kodebid 68: Opret forbindelse til database

Herefter oprettes et tomt array, som senere skal bruges til lagring af de resultater, som forespørgslen returnerer og brugerparametrene defineres som variable i php-scriptet, se Kodebid 69.

```
$search = array();

$lat = $_GET['lat'];
$lng = $_GET['lng'];
$radius = $_GET['radius'];
$type = $_GET['type'];
```

Kodebid 69: Bruger parametre

Herefter kan den egentlige analyse påbegyndes. Forespørgslen til databasen afhænger af, hvilket datalag brugeren har valgt at spørge til (i projektet enten *skoler* eller *fortidsminder*). Nedenstående kode viser, hvorledes det, ved brug af en `if`-statement, styres hvilket kald der skal sendes til databasen, se Kodebid 70.

```
if ($type == "skoler"){
      $post = pg_query($dbconn,
      "SELECT * FROM skoler WHERE ST_DWithin(ST_Transform(the_geom,
25832),ST_Transform(ST_GeomFromText('POINT($lng $lat)', 4326),
25832), $radius)");
      }
else{
      $post = pg_query($dbconn,
      "SELECT * FROM fortidsminder WHERE ST_DWithin(ST_
Transform(the_geom, 25832),ST_Transform(ST_GeomFromText('POINT($lng
$lat)', 4326), 25832), $radius)");
      };
while ($row =pg_fetch_object($post))  {
            $search[] = $row;
      };
```

Kodebid 70: Forspørgslen

Kaldet til databasen er et SQL-kald, som vælger alle rækker i tabellen *skoler* (eksempel), som ligger indenfor det brugerdefinerede punkts søgeradius. De valgte rækker indsættes herefter i arrayet "search", se Kodebid 70. Ovenstående SQL-kald er et rummeligt kald, hvor det har været nødvendigt at foretage yderligere konfiguration for at kunne eksekvere kaldet korrekt. Dette skyldes at applikationen benytter Googles referencesystem (WebMercator), mens databasens data ligger i UTM 32N ETRF89, hvilket er det gængse format til lagring af geodata i Danmark. Det er derfor nødvendigt at foretage en transformation af koordinaterne fra applikationen, så de har samme reference som databasens data. Dette gøres ved tilføjelse af kommandoen `ST_Transform`, hvorefter det ønskede SRID defineres.

Desuden er det også nødvendigt at præcisere, at variablerne `$lat` og `$lng` skal benyttes som geometritypen punkt og er defineret i SRID: 4326 (WebMercator). Dette gøres med nedenstående tilføjelse, se Kodebid 71.

```
ST_GeomFromText('POINT(($lat $lng))', 4326)
```

Kodebid 71: St_GeomFromText - point

Med ovenstående tilføjelser kan forespørgslen til databasen eksekveres, og resultatet sendes tilbage til applikationen. Dette gøres ved hjælp af nedenstående kode, som koder arrayet ved hjælp af JSON (JavaScript Object Notation) og sender det retur til applikationen, se Kodebid 72.

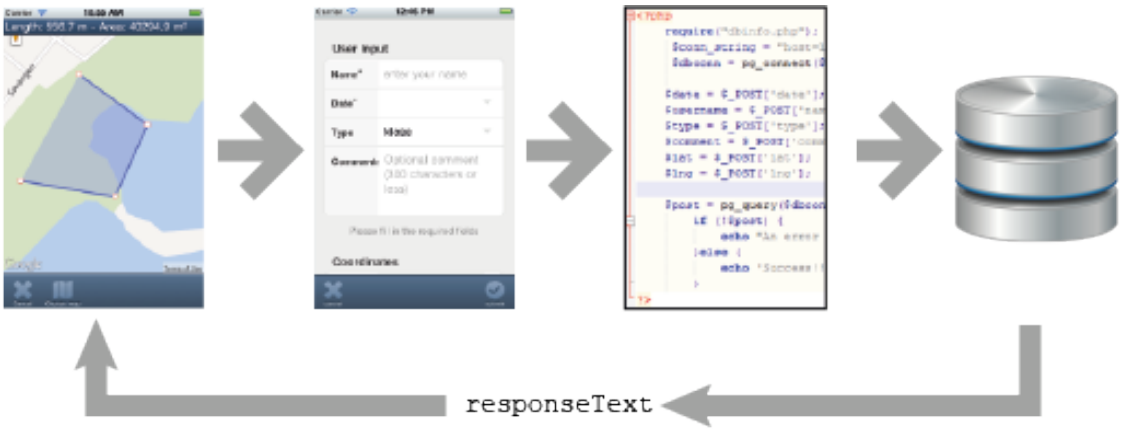
```
echo json_encode(array('search' =>$search));
```

Kodebid 72: Json encode

I applikationen indlæses resultatet af forespørgslen direkte til en datastore (searchstore) hvorfra listevisningen i trin 3 genereres, se Kodebid 72.

6.9 Tilføj data til databasen

Applikationens tre funktioner til tilføjelse af data til databasen(*add point*, *add line* og *add polygon*), fungerer alle på grundlæggende samme måde. En given geometri tegnes og gemmes i et array, hvorefter brugeren indtaster information om geometrien via en formular og endelig sendes arrayet til et PHP script der omformaterer det og sender det til databasen, se Figur 39.



Figur 39: Polygon til database

Hvordan geometrierne tegnes og gemmes i et array, er allerede blevet beskrevet i afsnit 6.7 *Målefunktioner*, og vil derfor ikke blive berørt mere her.

Submit formularen som brugeren udfylder, efter han/hun har tegnet en geometri, er opbygget af et Sencha Touche *formpanel* (`Ext.form.FormPanel`). Kodebid 73 viser en del af hvordan formularen for *Add polygon* er sat op.

```
var submitPolygonForm = new Ext.form.FormPanel({
      fullscreen: true,
      scroll: 'vertical',
      standardSubmit: false,
      id: "polygonForm",
      title: 'polygonForm',
      items: [{
            xtype: 'fieldset',
            title: 'User input',
            instructions: 'Please fill in the required fields',
            items: [
                  {
                  xtype: 'textfield',
                  type: 'int',
                  name: 'name',
                  label: 'Name',
                  placeHolder: "enter your name",
                  useClearIcon: false,
                  autoCapitalize: false,
                  required: true,
                  },
```

Kodebid 73: Measure objekt

Felterne i formularen er defineret som items, og kan være af forskellige typer(xtype) med diverse egenskaber(*fieldset*, *datepicker, selectfield* mfl.).

Når formularen er udfyldt, og brugeren trykker på submit knappen, køres en *handler* der sender de udfyldte parametre, sammen med geometriarrayet videre til et PHP script. Kodebid 74 viser denne handler.

```
var submitPolygonHandler = function(button, event) {
      Ext.Ajax.request({
      url : 'php/serverPolygon.php' ,
      params : {    name : submitPolygonForm.getValues().name,
                  type : submitPolygonForm.getValues().type,
                  date : submitPolygonForm.getValues().date,
                  comment : submitPolygonForm.getValues().comment,
                  array : submitPolygonForm.getValues().array,
            },
      method: 'POST',
            success: function ( result, request ){Ext.Msg.alert('Respond
from server', result.responseText, function(btn, evt){
                        window.location.reload();
                        });
                  },
            failure: function ( result, request) {
                        Ext.Msg.alert('Respond from server', result.
responseText, Ext.emptyFn);
      }
});
};
```

Kodebid 74: SubmitPolygonHandler

Handleren udfører et AJAX request(`Ext.Ajax.request`), der sender variable fra formularen(`submitPolygonForm.getValues()`) videre til et PHP script på serveren(`url: 'php/serverPolygon.php'`). Lykkedes det at sende variablene rigtigt af sted og efterfølgende tilføje dem til databasen, sendes et svar tilbage fra PHP scriptet, med beskeden *'success!!'*, og applikationen genindlæses(for at sikre at alle funktioner er nulstillet). Går der noget galt mellem PHP scriptet og databasen sendes et andet svar tilbage fra PHP scriptet, der beskriver fejlen. Kan handleren slet ikke sende noget til PHP scriptet, kommer der en advarsel med et tomt *"respond from server"*.

Det koordinatarray som PHP scriptet modtager fra handleren har følgende opbygning:

```
(lat1, lng2),(lat2, lat2),(lat3, lat3),…
```

Denne opbygning vil en PostGIS database i midlertidigt ikke kunne tolke som et polygon. Da PostGIS benytter OGCs konvention for *well known text representation*, der angiver at første- og sidste koordinatsæt skal være ens i et polygonarray, se nedenstående eksempel på et PostGIS polygonarray

```
(lng1,lat1,lng2,lat2,lng3,lat3, … ,lng1,lat1)
```

Bemærk også at et PostGIS array har hverken mellemrum eller parenteser mellem koordinatsættene og at breddegrad(lat) og længdegrad(lng) er omvendt i forhold til Googles MVCarray. Det er altså nødvendig at omformatere arrayet til OGCs *well known text* inden det sendes videre til databasen. Kodebid 75 viser hvordan dette opnås med PHP (se også Bilag 7).

```
$array = str_replace("(", "", $array);
$array = str_replace(")", "", $array);
$array = str_replace(" ", "", $array);

$explode = explode(",",$array);
$string = "$explode[0],$explode[1]";

$array = "$array,$string";

$array = explode(",",$array);

$i = 0;
$s= count($array);
$polygon= "";

for ($i=0;$i<=$s-1; $i++)  {
      $lat = $array[$i];
      $i++;
      $lng = $array[$i];

      $polygon= "$polygon,$lng $lat";
};

$polygon= substr($polygon,1);
$polygon= "($polygon)";
```

Kodebid 75: Submit Polygon PHP

Først fjernes parenteser og mellemrum (`str_replace`), så gemmes det første koordinatsæt som en variabel (`$string ="$explode[0], $explode[1]"`), og tilføjes i ende af arrayet (`$array = "$array,$string"`). Da dette nu er en lang tekst string, må vi igen kører en *explode*, for at omdanne variablen `$array` til et array igen. Der oprettes nu tre variable `$i`(der benyttes som tæller), `$s`(der indeholder arrayets længde) og `$polygon`(der skal indeholde det endelige polygonarray). Disse tre variable indgår i en *for*-løkke, der bytter om på `lat` og `lng` i arrayet og placerer dem i variablen `$polygon`. Endelig fjernes kommaet foran første koordinat i `$polygon (substr($polygon,1))` og der sættes en parentes omkring variablen `($polygon= "($polygon)";)`. Efter denne omformatering er arrayet klar til at blive sendt til databasen, se Kodebid 76.

```
$conn_string = "host=$ip port=5432 dbname=$database user=$username
password=$password";
$dbconn = pg_connect($conn_string);

$post = pg_query($dbconn, "INSERT INTO temp_paragraf3 (date,
username, objekttkst, comment, the_geom) VALUES ('$date',
'$username', '$type', '$comment', ST_Transform(ST_GeomFromText('MULT
IPOLYGON(($polygon))', 4326), 25832))");
      if (!$post) {
            echo "An error occured.";
      }else {
            echo 'Success!!!!!!!!';
```

Kodebid 76: Opret forbindelse og post geometri

Først oprettes der forbindelse til databasen(`pg_connect($conn_string);`), hvorefter et SQL-query benyttes til at sætte variablene og polygonarrayet ind i databasen. For at kunne sætte polygonarrayet rigtigt ind i databasen, må vi først omdanne dette til et PostGIS `ST_Geometry` objekt, og transformere dette til det rigtige referencesystem. `ST_GeomFromText` er en PostGIS specifik kommando, der opretter et PostGIS `ST_Geometry` objekt ud fra OGC *Well-Known text*, se Kodebid 77.

```
ST_GeomFromText('MULTIPOLYGON(($polygon))', 4326)
```

Kodebid 77: St_GeomFromText

Eftersom vores koordinater er angivet i *web mercator* projektionen, er vi nød til at transformere projektionen, som beskrevet i afsnit 6.8 *Rumlige forespørgelser.* Og endelig angiver koden, hvad der skal ske hvis det hhv. ikke lykkes eller lykkes at sende til database.

6.10 Signaturforklaring

Applikationens signaturforklaring findes i toolbaren yderst til højre, under *legend*. Opbygningen af denne er en simpel liste, der henter billeder via et `GetLegendGraphics` kald til GeoServers API, se Kodebid 78.

```
"http://localhost:8080/geoserver/wms?REQUEST=GetLegendGraphic&VERSIO
N=1.0.0&FORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=Peter:paragraf3"
```

Kodebid 78: GetLegendGraphic URL kald

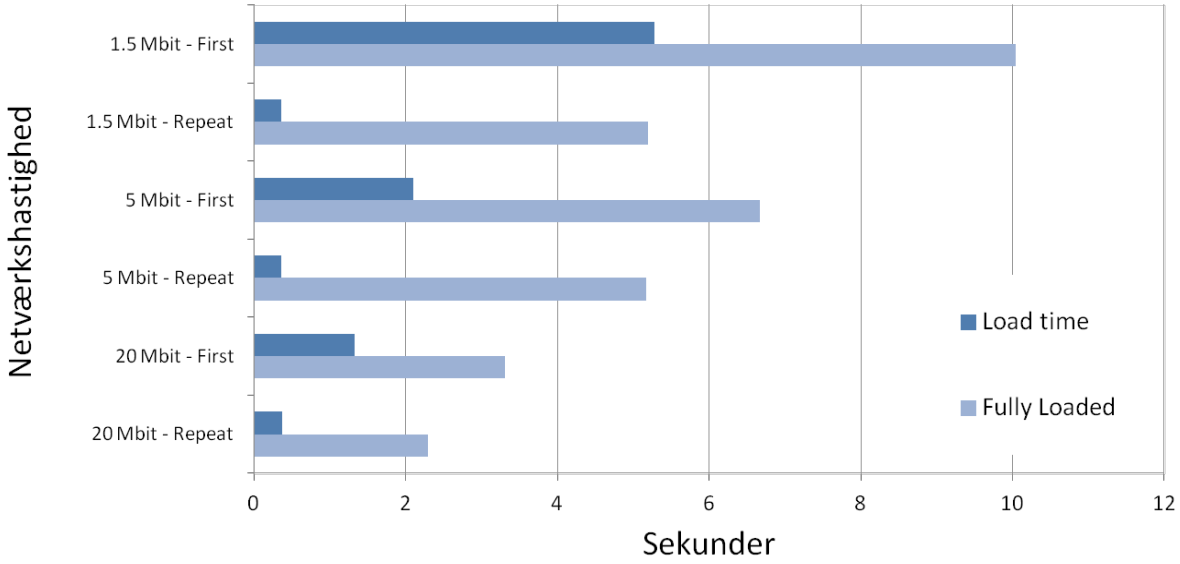
Som det ses defineres dimensionerne, lag og grafik type i selve URL'en. Mere information om GeoServers API kan findes på i GeoServers dokumentation. (Geoserver 1 2012).

7. Diskusion og test

7.1 Hastighedstest

Som berørt, i afsnit 5.11.6 *Hastighed og performance,* er det af afgørende betydning, at brugeren ikke skal vente ret længe på at applikationen indlæses. Det findes derfor naturligt, at teste hvorledes den udarbejdede applikation opfylder disse behov. Nedenstående figur viser resultatet af en sådan hastighedstest for projektapplikationen, se Figur 40.

Testen viser indlæsningshastigheden af projektapplikationen for tre forskellige netværkshastigheder. Der skelnes mellem førstegangsvisning (*First*) og genvisning (*Repeat*) af applikationen, ligesom der også skelnes mellem indlæsningstiden for hhv. selve applkationen (*Load time*) og indlæsning af både applikation og kortmateriale (*Fully Loaded*). Testen er lavet vha. onlineværktøjet *webpagetest.org*, hvor der er kørt ti testkørsler for hver netværkshastighed.

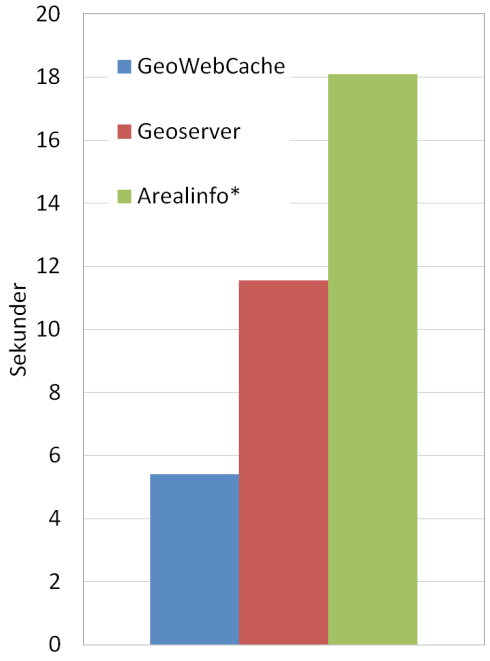


Figur 40: Test af indlæsningshastighed for projektapplikationen. Testen viser både tider for førstegangsvisning og genvisning af applikationen. Testen er udført vha. www.webpagetest.org

Fra tidligere, se afsnit 5.11.6, ved vi, at 25% af brugerne forlader en side, hvis den er mere end fire sekunder om at indlæse. Skal projektapplikationen opfylde disse krav, skal brugeren altså som udgangspunkt have en hurtig netværksforbindelse (≥ 20Mbit). Billedet er dog lidt mere nuanceret end som så. Brugeren kan nemlig starte brugen af applikationen allerede inden kortmaterialet er fuldt indlæst.

Det betyder reelt, at brugeren kan benytte applikationen allerede efter godt fem sekunder med den langsomste netværksforbindelse, og at hurtigere forbindelser indlæser applikationen på bare hhv. to og halvandet sekund, hvilket absolut må siges at være acceptabelt. Tiderne for genindlæsning resulterede heller ikke i lange ventetider for brugerene.

Selv på mobile netværk, må indlæsningshastigheden forventes at være acceptabelt, dog er der i projektapplikationen designet et skærmbillede som præsenteres for brugeren mens den resterende del af applikationen hentes. Herved kan brugeren se, at indlæsningen af applikationen er i gang, og er derved forhåbentligt mindre tilbøjelig til at forlade applikationen mens den indlæser.



Figur 41: WMS hastigheder *bemærk at Arealinfo ikke er fuldt indlæst

Som beskrevet i afsnit 6.4.1 *Geoserver*, gør projektapplikationen brug af GeoWebCache til prægenerering af tiles for yderligere at optimere hastigheden. For at teste effekten af dette laves en test, hvor projektapplikationen indlæses med samme overlay hentet fra hhv. GeoWebCache, Geoserver og arealinfo.dk. Ovenstående figur viser resultatet af denne test, se Figur 41.

Resultatet af testen viser tydeligt, hvor vigtig prægenerering af tiles er for hastigheden. Hastighedstest af webapplikationer er forbundet med en lang række usikkerheder, som eksempelvis netværksbelastning på testtidspunktet, hastighed af testeserveren mm. Det kan derfor være svært, at fastslå nogle eksakte indlæsningstider.

De udførte tests må dog forventes, at kunne give et rimeligt billede af projektapplikationens indlæsningshastighed.

Alt i alt må projektapplikationens performance siges, at være ganske acceptabel. Applikationen indlæses hurtigt, at fremstår hurtigreagerende og flydende i brug. Der er dog stadig mulighed for, at optimere applikationens performance yderligere. Dette kunne ske gennem optimering af Senchas standardkonfiguration, ligesom valg af en hybrid- eller lokalapplikationstype, jf afsnit 5.8 *Lokale, Web- og hybride applikationer*, ville kunne spare brugeren for ventetid.

Den konstante udvikling af de mobile enheders regnekraft og den hastigt stigende forbindelseshastighed, vil åbne helt nye muligheder for fremtidens kortapplikationer. Som vist i dette projekt, er det på nuværende tidspunkt nødvendigt, at outputte overlay kortmateriale vha. af en WMS-tjeneste og prægenererede tiles, for at opretholde en acceptabel applikationsafvikling. Dette kan allerede om få år være unødvendigt. Dermed åbnes for helt nye muligheder i forhold til arbejdet med geodata. En nærliggende mulighed ville være, at kortoverlays eksempelvis blev outputtet via en WFS-tjeneste, og at brugeren, i tillæg til den allerede eksisterende mulighed for at tilføje en ny geometri, ville have mulighed for at editere i en allerede eksisterende geometri.

7.2 Positionsbestemmelse

Som nævnt i teoriafsnittet 5.9 *Lokalisering,* er stedfæstelse af data alfa og omega i arbejdet med geodata. I den forbindelse er viden om stedsbestemmelsens nøjagtighed en yderst vigtig information, som ofte er afgørende for de videre anvendelsesmuligheder af data.

I projektapplikationen benyttes flere forskellige metoder til positionsbestemmelse, og det findes derfor naturligt, at undersøge hvor nøjagtige disse metoder kan forventes at være. Yderlige ønskes projektapplikationens positionsbestemmelsesværktøjer testet mod smartphonens eksisterende værktøjer.

I testen benyttes følgende positionsbestemmelsesværktøjer:

- ★ Garmin eTrex30, håndholdt outdoor-GPS med forbindelse til både GPS- og GLONASS-satellitter. GPS'en benyttes som reference og har under testen angivet sin nøjagtighed < 5 meter.
- ● iPhone lokalapplikation med ubegrænset adgang til smartphonens lokationsbestemmelsesværktøjer (test udført på iPhone 4. gen)
- ● Projektapplikationens "Locate"- funktion (test udført på iPhone 4. gen)
- ● Projektapplikationens "Get Coordinates"-funktion, hvor testpunktet placeres vha. touchskærm (test udført på iPad 2 gen)

Der opmåles i alt ni punkter liggende i varierende omgivelser (fritliggende, under bevoksning, langs og mellem høje bygninger, i hulveje etc.), for at give det mest realistiske testbillede. Punkterne er opmålt i området omkring Malmparken, Ballerup. Herefter beregnes den gennemsnitlige afvigelse fra reference-GPS'ens koordinater, se Tabel 5 (se også bilag 10 *Test, Positionsbestemmelse*).

Værktøj	Min. afvigelse	Max. afvigelse	Gns. afvigelse
iPhone lokalapplikation	3,5 m	22,2 m	10,0 m
App "Locate"	2,6 m	35,2 m	14,7 m
App "Get Coordinates"	1,5 m	13,1 m	6,7 m

Tabel 5: Opmålinger

Ovenstående testresultater, stemmer fint overens med nøjagtighederne beskrevet i afsnit 5.9 *Lokalisering*. Den lokale iPhone-applikation, med fuld adgang til enhedens GPS funktionalitet, ligger som forventet med en gennemsnitlig afvigelse på 10 meter. Mere overraskende er resultatet mht. til projektapplikationens "*Locate-funktion*". Funktionen har ikke fuld adgang til enheden, og bygger derfor på en positionsberegning ud fra netværksmaster og WPS. Nøjagtigheden vil derfor afhænge meget af, tilstedeværelsen af trådløse netværk. Testen viser dog en noget bedre nøjagtighed end forventet.

Endeligt kommer turen til projektapplikationens "Get Coordinates", som er udviklet ud fra tesen om, at det, med et ortofoto som hjælp, er muligt at udpege en ønsket position ganske præcist vha. projektapplikationens touch interface. Resultatet af testen viser da også, at denne tese holder stik. Med en gennemsnitlig afvigelse på bare 6,7 meter, må denne metode siges at være fuldt på højde med registrering vha. GPS-signal. Testen viser også, at der må forventes en vis spredning/usikkerhed på de returnerede koordinater.

Nedenstående figurer viser eksempler på, hvorledes nogle resultater ligger nogenlunde pænt, se Figur 42, mens andre afviger meget fra referencepunktet, se Figur 43.

Figur 42: Testpunkt 3 - god nøjagtighed

Figur 43: Testpunkt 2 - Possistions-bestemmelsen vha. Locate-funktion afviger fra referencepunktet med 35m

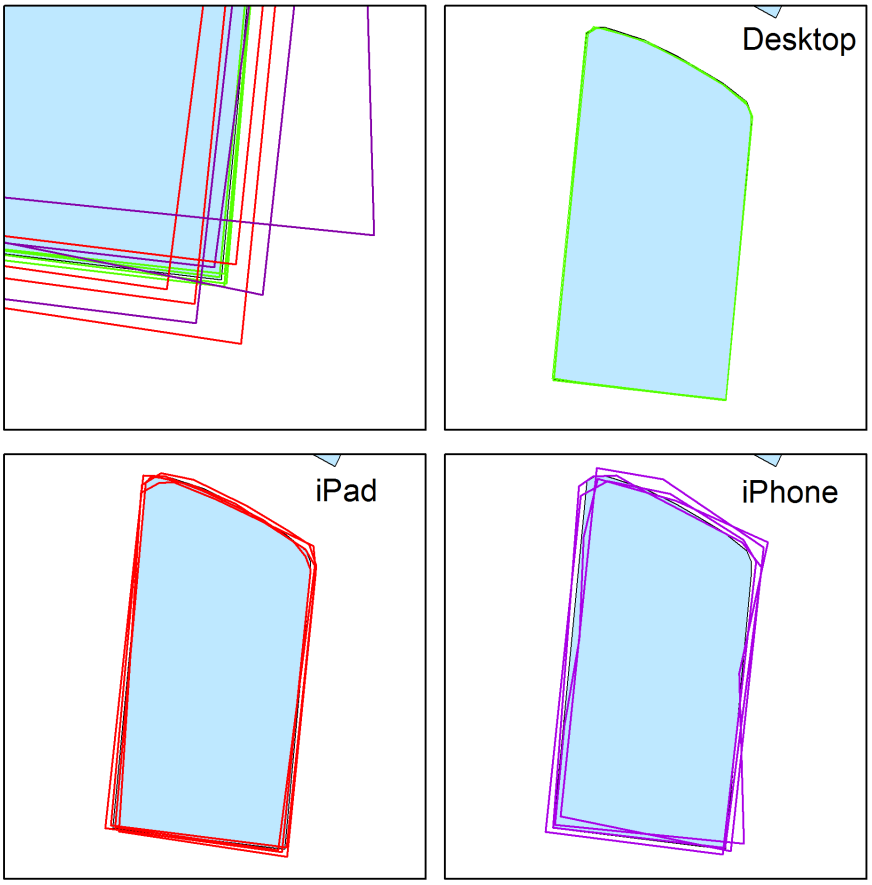
Afvigelserne i testen kan i høj grad tilskrives de begrænsninger de benyttede positionsbestemmelsesværktøjer har, samt det begrænsede antal testpunkter. Derudover må usikkerheden på referencemålingen også tages i mente. Selvom nøjagtigheden for referencemålingen angives til at ligge under fem meter, må testens resultat ses som en antydning af nogle tendenser, frem for et egentligt facit.

Testens begrænsninger ændrer dog ikke på, at projektapplikationens Locate-funktion giver et bedre resultat end forventet, samt at metoden til dataindberetning vha. af touch-skærmen, viser sig som en god og præcis indberetningsmetode, sammenlignet med de alternativer der findes på den mobile platform.

7.3 Indregistreringstest

For at få et billede af hvor god en præcision der kan forventes, når der tilføjes nye geometrier til databasen, blev der foretaget en test af *Add polygon* funktionaliteten.

I testen blev fire personer, uafhængigt af hinanden, sat til at tilføje det samme område (Sankt Jørgens Sø i København) til databasen tre gange på tre forskellige enheder. Hhv. en desktop maskine(med Google Chrome webbrowser og en mus), en iPad(2.gen) og en iPhone(4.gen), se Figur 44.



Figur 44: Øverst tv. Forstørrelse af samtlige indregistreringer, Øverst th. Desktop indregistreringer, Nederst tv. iPad indregistreringer og Nederst th iPhone indregistreringer

Efterfølgende opmålte vi afstanden fra søens nederste højre hjørne og ud til det hjørne testpersonerne havde sat, se Figur 44 øverst tv.. Nedenstående beregninger viser den gennemsnitlige afvigelse på det pågældende hjørne.

$$Desktop = \frac{0{,}46m + 0{,}85m + 0{,}83m + 0{,}76m}{4} = 0{,}7\ meter$$

$$iPad = \frac{2{,}59m + 4{,}37m + 6{,}79m + 8{,}20m}{4} = 5{,}5\ meter$$

$$iPhone = \frac{1{,}78m + 6{,}17m + 5{,}40m + 19{,}51m}{4} = 8{,}2\ meter$$

Som det fremgår, er der en markant forskel på hvor god en præcision, der opnås på de forskellige enheder. Ikke overraskende giver desktop registreringerne de suverænt bedste resultater, mens præcisionen falder drastisk i takt med at skærmstørrelsen bliver mindre og indtastningen forgår med fingrene. Overordnet set må præcisionen på touch enhederne siges at være noget skuffende. En stylus kunne muligvis forbedre præcisionen på touch enheder, da særligt brugere med store fingre har problemer med at placere punkter præcist. Ligeledes ville større ikoner for polygonernes knudepunkter gøre det lettere for brugeren at rykke rundt på de allerede placerede punkter.

Bemærke at denne test ikke er videnskabeligt valid, til vurdering af applikationens præcision, da det empiriske grundlag ikke er tilstrækkeligt. Derfor skal testen også kun ses som et praj, om den forventede præcision ved tilføjelse af data.

Det data der arbejdes med i applikationen er særligt paragraf-3 områder (beskyttede naturtyper). For at kunne vurdere hvorvidt præcisionen er af tilstrækkelig, må vi derfor se lidt nærmere på hvilken kvalitet eksisterende paragraf-3 data har. Basis registrering af paragraf-3 områder er foretaget ud fra luftfotos, og indeholder derfor en del fejl og usikkerheder, da det kan være svært at skelne mellem forskellige naturtyper ud fra luftfotos(Jysk Landbrugsrådgivning n.d.). Da paragraf-3 områder i høj grad består af vild natur, vil der ligeledes være en del *fuzzyness* i data, da man ikke kan sætte en skarp linje på fx hvor en sø slutter og en mose starter.

Ud fra ovenstående må applikationens præcision vurderes til at være tilstrækkelig til indregistrering visse typer data(fx moser, søer, skove mv.), mens præcisionen er mere tvivlsom ift. ikke fuzzy data, hvor præcisionskravene er højere.

8. Konklusion

I den forhåndenværende opgave søgtes det klarlagt, i hvilken grad en mobilapplikation baseret på webstandarder og Open Source produkter vil kunne opfylde kravene for registrering og opdatering af paragraf-3 områder. Ud fra rapportens resultater mener vi at kunne konkludere, at en Open Source webapplikationen i høj grad vil kunne opfylde de behov / krav der er til en sådan applikation.

Indrapportering af rumligt data i applikationen konkluderes ligeledes at være tilfredsstillende i situationer, hvor præcisionskravene er relativt lave. Der findes dog tilfælde hvor præcisionskravene er for høje, og hvor der vil være behov for forbedringer, som fx bedre GPS tilgængelighed, bedre brugergrænseflade (med højere nøjagtighed fra evt. Stylus) samt en bredere udbredelse af webstandarder som Webkit og HTML5 .

I opgaven blev det også undersøgt hvilke faktorer, der er af betydningen for applikationens overordnede performance. Ud fra resultaterne beskrevet i bl.a. afsnit 7.1 *Hastighedstest*, må det konkluderes, at en webapplikation udemærket kan have en tilfredsstillende performance, såfremt der arbejdes målrettet med optimering af hastigheden, ved fx at bruge prægenerede tiles (cached WMS) og en effektivt indekseret rumlig database.

Det må konkluderes, at der er både fordele og ulemper ved den mobilplatform ift. digital forvaltning. Den mobileplatform gør det muligt at registrere data i felten direkte på en tablet eller en telefon. Herved åbnes op for at folk, der ikke er GIS eksperter, kan indregistrere data, hvorved et led i indregistreringsprocessen kan skæres bort.

Med udgangspunkt i de erfaringer der er gjort gennem udviklingen af projektapplikationen, må det konkluderes, at en mobil applikation skal have et simpelt layout, der tager højde for skærmstørrelsen og touch-brugerfladen. Ligeledes konkluderes det, at en mobil applikation overordnet set skal designes til et relativt fokuseret og specialiseret anvendelsesområde.

I opgaven er der arbejdet meget med at gøre applikationen platformsuafhængig. Dette er til dels lykkedes, men det må og konkluderes at der endnu er et stykke vej, førend webstandarder som Webkit og HTML5 er udbredte nok til at kunne kører ens på alle de store platforme.

9. Litteraturliste

Agile Alliance (n.d.). The Twelve Principles of Agile Software. [Online], Available at: http://www.agilemanifesto.org/principles.html [Accessed Maj 2012].

Allardice, S., (2011). JavaScript Essential Training. [Online], Available at: http://www.lynda.com/JavaScript-tutorials/Essential-Training-2011/81266-2.html, [Accessed Maj 2012];

Alizadeh, F. (2008). Opportunistic and Hybrid Localization, Worcester, MA, USA: Worcester Polytechnic Institute

Anderson, B. & Deoliveira, J., (2007) WMS Performance Tests! - Mapserver & Geoserver. Victoria, Canada, Refractions Research

Apple Inc., (2012). iOS Human Interface Guidelines, Cupertino: Apple Inc.;

Arbejdsgruppe om IT-arkitektur i regi af Det Koordinerende Informationsudvalg, (2003). Hvidbog om IT-arkitektur, København: Ministeriet for Videnskab, Teknologi og Udvikling;

Bachl, S., Tomitsch, M., Wimmer, C. & Grechenig, T., (2010). Challenges for Designing the User Experience of Multi-touch Interfaces, Berlin: ACM SIGCHI Symposium on Engineering Interactive Computing Systems;

Balstrøm, T., Jacobi, O. & Bodum, L., (2006). Bogen om GIS og geodata. 1. udgave ed. København: GIS & Geodata;

Batty, P., (2010). The geospatial revolution. Århus: Geoforum Danmark, Kortdage;

BBC, (2011). Google Maps to charge for usage. [Online], Available at: http://www.bbc.co.uk/news/business-15523050, [Accessed Maj 2012];

BindApple (2008). iPhone Keyboard [Online], Available at: http://bindapple.com/iphone-keyboard/ [Accessed Maj 2012]

Brutlag, J., (2009). Web Search Infrastructure: Speed Matters. [Online], Available at: http://googleresearch.blogspot.com/2009/06/speed-matters.html, [Accessed Maj 2012];

Cavazza, F., (2011). Mobile Web App vs. Native App? It's Complicated. [Online] , Available at: http://www.forbes.com/sites/fredcavazza/2011/09/27/mobile-web-app-vs-native-app-its-complicated/, [Accessed Maj 2012];

Cote, P., (n.d). Cultivating Spatial Intelligence - Understanding GIS Data, Referencing Systems and Metadata. [Online], Available at: http://internal.gsd.harvard.edu/gis/manual/data_basics/, [Accessed 9. Maj 2012];

Danmarks Statestik (2012). Statestikbanken [Online], Available at: http://www.statistikbanken.dk/, [Accessed Maj 2012]

DenOffentligeSektor.dk (2011). Giv et praj fra mobilen[Online], Available at: http://www.denoffentligesektor.dk/blog/giv-et-praj-fra-mobilen [Accessed Maj 2012]

Department of Economic and Social Affairs, (2012). E-Government Survey 2012 - E-Government for the People, New York: United Nations;

DMU (n.d). Miljø Ordbog [Online], Available at: http://www2.dmu.dk/1_om_DMU/2_akt-proj/ordspec.asp?ID=688, [Accessed maj 2012

ESRI, (2008). What is raster data?. [Online], Available at: http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=What_is_raster_data%3F, [Accessed 4. Maj 2012];

Fajstrup, L., (2006). Kortprojektioner og forvanskninger, Aalborg: Institut for Matematiske Fag, Aalborg Universitet;

Geoserver, (2012). WFS reference. [Online], Available at: http://docs.geoserver.org/latest/en/user/services/wfs/reference.html#wfs-reference, [Accessed Maj 2012];

Geodata-info (n.d). [Online], Available at: http://www.geodata-info.dk/portal/about.aspx [Accessed maj 2012]

Geoserver 1, (2012). WFS reference. [Online], Available at: http://docs.geoserver.org/latest/en/user/services/wms/get_legend_graphic/legendgraphic.html, [Accessed Maj 2012];

Hansen, H. S., Schøder, L., Hvingel, L. & Jesper, C. S., (2010). Er det offentlige Danmark parat til geografisk baseret digital forvaltning?. Perspektiv, Volume 18;

Heywood, I., Cornelius, S. & Carver, S., (2006). An Itroduction to Geographical Information Systems. 3. udgave ed. New Jersey: Prentice Hall;

Hird, J., (2011). The fight gets technical: mobile apps vs. mobile sites. [Online], Available at: http://econsultancy.com/uk/blog/7832-the-fight-gets-technical-mobile-apps-vs-mobile-sites, [Accessed Maj 2012];

Hvingel, L. & Hansen, H. S., (2011). Geodata som katalysator for digital forvaltning. Perspektiv, Volume 20;

Kaneda, D., (2010). An Introduction to Theming Sencha Touch. [Online], Available at: http://www.sencha.com/blog/an-introduction-to-theming-sencha-touch, [Accessed Maj 2012];

KMD, (2012). Den Offentlige sektor.dk. [Online], Available at: http://www.denoffentligesektor.dk/pressemeddelelser/vindere-af-digitaliseringsprisen-2012-er-fundet, [Accessed 2 Maj 2012];

Kokkendorff, S. L., (2010). Kortforsyningsseminar - Nyt om projektioner, København: Kort & Matrikelstyrelsen;

Jysk Landbrugsrådgivning (n.d.) [Online]- http://www.jlbr.dk/Raadgivning/Deltidsraadgivning/NaturOgMiljoe/Naturbeskyttelse.htm [Accessed Maj 2012]

Longley, P. A., Goodchild, M., Maguire, D. J. & Rhind, D. W., (2004). Geographic Information Systems And Science. 1. udgave ed. New York: Wiley;

Michal, (2011). Skyhook WPS location system reviewed. [Online], Available at: http://www.articlesbase.com/cell-phones-articles/skyhook-wps-location-system-reviewed-5036638.html, [Accessed Maj 2012];

Microsoft, (2012). Bing Maps Tile System. [Online], Available at: http://msdn.microsoft.com/en-us/library/bb259689.aspx, [Accessed Maj 2012];

Miljøministeriet, (2010). Aftale mellem Miljøministeriet og KL om et bedre grundlag for beskyttelse af værdifuld dansk natur, København: Miljøministeriet;

Open Geospatial Consortium Inc., (2006). OpenGIS® Web Map Server Implementation Specification, s.l.: Open Geospatial Consortium Inc.;

Pearce J. (2011) Idiomatic Layouts with Sencha Touch. [Online], Available at: http://www.sencha.com/learn/idiomatic-layouts-with-sencha-touch/, [Accessed maj 2012]

Regeringen, KL og Danske Regioner, (2011). Den digitale vej til fremtidens velfærd - Den fællesoffentlige digitaliseringsstrategi 2011-2015, København: Økonomistyrelsen;

Robinson, J., (2010). The HTML5 Family: CSS3. [Online], Available at: http://www.sencha.com/blog/the-html5-family-css3, [Accessed Maj 2012];

Sachs, J., (2009). API vs. SDK. [Online], Available at: http://stackoverflow.com/questions/834763/api-vs-sdk, [Accessed Maj 2012];

Sanford, J., (2011) Line Length and Polygon Area With Google Maps API V3 [Online], Available at : http://geojason.info/demos/line-length-polygon-area-google-maps-v3/ [Accessed mah 2012]

Seib, C. & Florin, C., (2003). American National Standards Institute. X3H2 Records, 1978-1995. [Online], Available at: http://special.lib.umn.edu/findaid/xml/cbi00168.xml, [Accessed April 2012];

Sencha (n.d). Sencha Touch v.1.1 Documentation [Online], Available at: http://docs.sencha.com/touch/2-0/#!/api/Ext.util.Geolocation , [Accessed Maj 2012]

Sights, (2012). browsers. [Online], Available at: http://html5test.com/compare/browser/index.html, [Accessed Maj 2012];

Skyhook, (2012). Hybrid Positioning System (XPS). [Online], Available at: http://www.skyhookwireless.com/howitworks/xps.php, [Accessed Maj 2012];

StatCounter, (2012). Global Stats. [Online], Available at: http://gs.statcounter.com/, [Accessed maj 2012];

Statistic Brain, (2012). Google Annual Search Statistics. [Online], Available at: http://www.statisticbrain.com/google-searches/, [Accessed 14 April 2012];

Teknologirådet, (2005). Delrapport i Teknologirådets projekt om ”Privatlivets fred i digital forvaltning", København: Teknologirådet;

w3schools.com, (n.d.) CSS Introduction. [Online], Available at: http://www.w3schools.com/css/css_intro.asp, [Accessed Maj 2012];

Wang, F. & Ren, X., (2009). Empirical Evaluation for Finger Input Properties In Multi-touch Interaction, Boston: Deptartment of Information Systems Engineering, Kochi University of Technology;

Wang, S., Min, J. & Yi, B. K., (n.d.) Location Based Services for Mobiles: Technologies and Standards, s.l.: LG Electronics Mobile Research;

Worklight®, (n.d.) HTML5, Hybrid or Native Mobile App Development, New York: IBM;

World Wide Web Consortium 1, (n.d.) HTML/Training/Basic content. [Online], Available at: http://www.w3.org/wiki/HTML/Training/Basic_content, [Accessed Maj 2012];

World Wide Web Consortium 2, (n.d.) The Web and Mobile Devices. [Online], Available at: http://www.w3.org/Mobile/, [Accessed Maj 2012];

World Wide Web Consortium 3, (n.d.) Geolocation API Specification. [Online], Available at: http://dev.w3.org/geo/api/spec-source.html, [Accessed Maj 2012];

World Wide Web Consortium 4, (n.d.) CSS intro [Online], Available at: http://www.w3schools.com/css/css_intro.asp [Accessed Maj 2012]

Zandbergen, P. A., (2009). Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning. Transactions in GIS, Issue 13, pp. 5-26;

10. Bilag

Bilag 1: index.html

Bilag 2: index.js

Bilag 3: funktioner.js

Bilag 4: dbinfo.php

Bilag 5: server.php

Bilag 6: serverLine.php

Bilag 7: serverPolygon.php

Bilag 8: serverRadius.php

Bilag 9: ST_intersect.php

Bilag 10: Test, Positionsbestemmelse

Bilag 1: index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Alfred</title>

    <link rel="stylesheet" href="touch/resources/css/sencha-touch.css" type=
"text/css">
    <script type="text/javascript" src=
"http://maps.google.com/maps/api/js?sensor=true&libraries=geometry"
></script></script>
    <script type="text/javascript" src="touch/sencha-touch.js"></script>
    <script type="text/javascript" src=
"http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
    <script type="text/javascript" src="js/slider.js"></script>
    <script type="text/javascript" src="js/funktioner.js"></script>
    <script type="text/javascript" src="js/index.js"></script>


</head>
<body>
</body>
</html>
```

```
//-----------------------------------------------------------------------------
//  Variabler til lagring af data
//-----------------------------------------------------------------------------

    var mapPanel;
    var positionArr = ExtMap.Storage.loadPreviousPosition();
    var markersArray = []; //Dette er et array der indholder alle markers
    var markerLat;
    var markerLng;
    var length;
    var area;
    var resultvar;
    var search;
    var opacity;



    var measure = {
            mvcLine: new google.maps.MVCArray(),
            mvcPolygon: new google.maps.MVCArray(),
            mvcMarkers: new google.maps.MVCArray(),
            line: null,
            polygon: null
        };

//-----------------------------------------------------------------------------
//  Vinyl Fox quick fix af fejl vedr. enableHighAccuracy til GPS lokation
//-----------------------------------------------------------------------------

Ext.override(Ext.util.GeoLocation,{
    parseOptions: function(){
        var ret = {
            maximumAge: this.maximumAge,
            enableHighAccuracy: this.allowHighAccuracy
        };
        //Google doesn't like Infinity
        if(this.timeout !== Infinity){
            ret.timeout = this.timeout;
        }
        return ret;
    }
});

//-----------------------------------------------------------------------------
//  Handlers
//-----------------------------------------------------------------------------

    var tapHandler = function(button, event) {
            Panel.setActiveItem(signaturCard, {type: 'slide'});
        };

    var submitCardHandler = function(button, event) {
                Panel.setActiveItem(submitCard, {type: 'slide'});
```

```
                submitForm.setValues({lat: markerLat, lng: markerLng});

          };

    var submitPolyCardHandler = function(button, event) {
                Ext.getCmp('infoBarId').hide();
                Panel.setActiveItem(submitPolygonCard, {type: 'slide'});
                submitPolygonForm.setValues({array: measure.mvcPolygon.getArray()});

          };

    var submitLineCardHandler = function(button, event) {
                Ext.getCmp('infoBar1Id').hide();
                Panel.setActiveItem(submitLineCard, {type: 'slide'});
                submitLineForm.setValues({array: measure.mvcPolygon.getArray()});

          };


    var submitProxCardHandler = function(button, event) {
                Panel.setActiveItem(submitProxCard, {type: 'slide'});
                submitProxForm.setValues({lat: markerLat, lng: markerLng});

          };


    var submitWithinCardHandler = function(button, event) {
                Ext.getCmp('infoBarId').hide();
                Panel.setActiveItem(submitWithinCard, {type: 'slide'});
                submitWithinForm.setValues({array: measure.mvcPolygon.getArray()});

          };


    var submitHandler = function(button, event) {
                    Ext.Ajax.request({
                    url : 'php/server.php' ,
                    params : { name : submitForm.getValues().name,
                                    type : submitForm.getValues().type,
                                    date : submitForm.getValues().date,
                                    comment : submitForm.getValues().comment,
                                    lat : submitForm.getValues().lat,
                                    lng : submitForm.getValues().lng,

                                },
                    method: 'POST',
                        success: function ( result, request ){Ext.Msg.alert(
'Respond from server', result.responseText, function(btn, evt){
                                    window.location.reload();
                                    })
                                    ;
                                    //console.log('Success', 'Data return from the
server: '+ result.responseText);
```

```
                                },
                                failure: function ( result, request) {
                                    Ext.Msg.alert('Respond from server', result.
responseText, Ext.emptyFn);
                                    //console.log('Failed', result.responseText);
                                }

                });
        };

    var submitProxHandler = function(button, event) {

                    ProxType = submitProxForm.getValues().type,
                    ProxRadius =  submitProxForm.getValues().radius,
                    ProxLat = submitProxForm.getValues().lat,
                    ProxLng = submitProxForm.getValues().lng,

                    ProxUrl = '/29feb/php/serverRadius.php?lat='+ProxLat+'&lng='+
ProxLng+'&type='+ProxType+'&radius='+ProxRadius,

                    console.log(ProxUrl);
                    searchStore.proxy.url = ProxUrl;
                    searchStore.load();
                    Panel.setActiveItem(proxListPanel, {type: 'slide', direction:
'left'})
                    Ext.getCmp('locateid').hide();
                    Ext.getCmp('toolid').hide();
                    Ext.getCmp('submitPolyId').hide();
                    Ext.getCmp('signaturid').hide();
                    Ext.getCmp('addid').hide();
                    Ext.getCmp('submitid').hide();
                    Ext.getCmp('submitProxId').hide();
                    Ext.getCmp('spacerid').hide();
                    Ext.getCmp('cancelid').hide();
                    Ext.getCmp('choosemapid').hide();
                    Ext.getCmp('spacerid').show();
        };



    var submitPolygonHandler = function(button, event) {
                    Ext.Ajax.request({
                    url : 'php/serverPolygon.php' ,
                    params : { name : submitPolygonForm.getValues().name,
                                    type : submitPolygonForm.getValues().type,
                                    date : submitPolygonForm.getValues().date,
                                    comment : submitPolygonForm.getValues().comment,
                                    array : submitPolygonForm.getValues().array,

                                },
                    method: 'POST',
                        success: function ( result, request ){Ext.Msg.alert(
'Respond from server', result.responseText, function(btn, evt){
```

```
                                    window.location.reload();
                                    })
                                    ;
                                    //console.log('Success', 'Data return from the
server: '+ result.responseText);
                                },
                                failure: function ( result, request) {
                                    Ext.Msg.alert('Respond from server', result.
responseText, Ext.emptyFn);
                                    //console.log('Failed', result.responseText);
                                }

                });
        };


    var submitWithinHandler = function(button, event) {

                    ProxType = submitWithinForm.getValues().type,
                    ProxArray =  submitWithinForm.getValues().array,

                    ProxUrl = '/29feb/php/st_intersect.php?type='+ProxType+'&array='
+ProxArray,

                    console.log(ProxUrl);
                    searchStore.proxy.url = ProxUrl;
                    searchStore.load();
                    Panel.setActiveItem(proxListPanel, {type: 'slide', direction:
'left'})
                    Ext.getCmp('locateid').hide();
                    Ext.getCmp('toolid').hide();
                    Ext.getCmp('submitPolyId').hide();
                    Ext.getCmp('signaturid').hide();
                    Ext.getCmp('addid').hide();
                    Ext.getCmp('submitid').hide();
                    Ext.getCmp('submitProxId').hide();
                    Ext.getCmp('spacerid').hide();
                    Ext.getCmp('cancelid').hide();
                    Ext.getCmp('choosemapid').hide();
                    Ext.getCmp('spacerid').show();
        };



        var submitLineHandler = function(button, event) {
                    Ext.Ajax.request({
                    url : 'php/serverLine.php' ,
                    params : { name : submitLineForm.getValues().name,
                                    type : submitLineForm.getValues().type,
                                    date : submitLineForm.getValues().date,
                                    comment : submitLineForm.getValues().comment,
                                    array : submitLineForm.getValues().array,
```

```
                                },
                    method: 'POST',
                        success: function ( result, request ){Ext.Msg.alert(
'Respond from server', result.responseText, function(btn, evt){
                                    window.location.reload();
                                    })
                                    ;
                                    //console.log('Success', 'Data return from the
server: '+ result.responseText);
                                },
                                failure: function ( result, request) {
                                    Ext.Msg.alert('Respond from server', result.
responseText, Ext.emptyFn);
                                    //console.log('Failed', result.responseText);
                                }

                });
        };

    var clear = function clearOverlays() {  // Dette er en funktion der clear markers
        if (markersArray) {
            for (i in markersArray) {
                    markersArray[i].setMap(null);
                    }
            }
            clearPolygon();
        };

    var clearAll = function(button, event){
        clearPolygon();
        clear();
    };

    var cancelHandler =  function(button, event) {
            Ext.getCmp('locateid').show();
            Ext.getCmp('toolid').show();
            Ext.getCmp('choosemapid').show();
            Ext.getCmp('signaturid').show();
            Ext.getCmp('addid').show();
            Ext.getCmp('submitid').hide();
            Ext.getCmp('submitPolyId').hide();
            Ext.getCmp('spacerid').hide();
            Ext.getCmp('cancelid').hide();
            Ext.getCmp('infoBarId').hide();
            Ext.getCmp('infoBar1Id').hide();
            Ext.getCmp('submitProxId').hide();
            Ext.getCmp('submitLineId').hide();
            google.maps.event.clearListeners(mapdemo.map, 'mousedown') ;
            clearAll()
        };

    var backHandler = function(button, event, typex) {
           Panel.setActiveItem(mapPanel, {type: 'slide', direction: 'right'});
```

```
            Ext.getCmp('locateid').show();
            Ext.getCmp('toolid').show();
            Ext.getCmp('choosemapid').show();
            Ext.getCmp('signaturid').show();
            Ext.getCmp('addid').show();
            Ext.getCmp('submitid').hide();
            Ext.getCmp('submitPolyId').hide();
            Ext.getCmp('spacerid').hide();
            Ext.getCmp('cancelid').hide();
            Ext.getCmp('infoBarId').hide();
            Ext.getCmp('infoBar1Id').hide();
            Ext.getCmp('submitProxId').hide();
            Ext.getCmp('submitLineId').hide();
            Ext.getCmp('submitLineCardId').hide();
            Ext.getCmp('submitLineCardId').hide();
            Ext.getCmp('submitLineCardId').hide();
            Ext.getCmp('submitWithinId').hide();
            google.maps.event.clearListeners(mapdemo.map, 'mousedown') ;
            clearAll()
        };

    var backToListHandler = function(button, event, typex) {
            Panel.setActiveItem(proxListPanel, {type: 'slide', direction: 'right'});
        };




//---------------------------------------------------------------------------
//  Signaturforklaring: data store og listevisning defineres
//---------------------------------------------------------------------------

    var signatur_store = new Ext.data.Store({
        model: Ext.regModel(' ', {
            fields: [
                {name:'id', type: 'int'},
                {name:'mapTypeName', type: 'string'},
                ]
            }),

        data: [
            {id: 1, mapTypeName: 'Polygons<br /><img style="opacity: 0.5;"
src="http://2.107.248.183:8080/geoserver/wms?REQUEST=GetLegendGraphic&VERSION=1.0.0&F
ORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=cite:paragraf3"/>'},
            {id: 2, mapTypeName: 'Points<br /><img style="opacity: 0.5;"
src="http://2.107.248.183:8080/geoserver/wms?REQUEST=GetLegendGraphic&VERSION=1.0.0&F
ORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=cite:fortidsminder"/><span
style="color:gray;font-size:0.7em;">Fortidsminder</span><br /><img style="opacity:
0.5;"
src="http://2.107.248.183:8080/geoserver/wms?REQUEST=GetLegendGraphic&VERSION=1.0.0&F
ORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=cite:skoler"/><span
style="color:gray;font-size:0.7em;">Skoler</span>'},
            {id: 3, mapTypeName: 'Lines<br /><img style="opacity: 0.5;"
```

```
src="http://2.107.248.183:8080/geoserver/wms?REQUEST=GetLegendGraphic&VERSION=1.0.0&F
ORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=cite:beskyt_linje"/>'},

            ]
    })

    var signatur_list = new Ext.List ({
        store: signatur_store,
        itemTpl: '{mapTypeName}',
        allowDeselect: false,
        singleSelect: true
        });

//-------------------------------------------------------------------------------
//  Submitform - PUNKT
//-------------------------------------------------------------------------------
    var submitForm = new Ext.form.FormPanel({
            fullscreen: true,
            scroll: 'vertical',
            standardSubmit: false,
            id: "form",
            title: 'form',
            items: [{
                xtype: 'fieldset',
                title: 'User input',
                instructions: 'Please fill in the required fields',
                items: [
                    {
                    xtype: 'textfield',
                    type: 'int',
                    name: 'name',
                    label: 'Name',
                    placeHolder: "enter your name",
                    useClearIcon: false,
                    autoCapitalize: false,
                    required: true,
                    },
                    {
                    xtype: 'datepickerfield',
                    slotOrder:
                                ['year', 'months', 'day']
                            ,
                    name: 'date',
                    label: 'Date',
                    useClearIcon: true,
                    autoCapitalize: false,
                    required: true
                    },
                    {
                    xtype: 'selectfield',
                    name: 'type',
                    label: 'Type',
                    options: [{
```

```
                                text: 'Fortidsminde', value: 'Fredet fortidspunkt'
                                }]
                    },
                    {
                    xtype: 'textareafield',
                    name: 'comment',
                    label: 'Comments',
                    placeHolder: "Optional comment (300 characters or less)",
                    useClearIcon: false,
                    autoCapitalize: false,
                    required: false,
                    maxLength: 300
                    },
                ],
                },
                {
                    xtype: 'fieldset',
                    title: 'Coordinates',
                    instructions: 'Marker coordinates',
                    items: [
                        {
                        xtype: 'textfield',
                        name: 'lat',
                        label: 'Latitude',
                        useClearIcon: true,
                        autoCapitalize: false,
                        required: false,
                        disabled: true
                        }, {
                        xtype: 'textfield',
                        name: 'lng',
                        label: 'Longitude',
                        useClearIcon: true,
                        autoCapitalize: false,
                        required: false,
                        disabled: true
                        }
                    ],
            }],
        });

//---------------------------------------------------------------------------
//  SubmitProxForm - Proximity
//---------------------------------------------------------------------------
    var submitProxForm = new Ext.form.FormPanel({
            fullscreen: true,
            scroll: 'vertical',
            standardSubmit: false,
            id: "proxForm",
            title: 'Proximity form',
            items: [{
                xtype: 'fieldset',
                title: 'User input',
```

```
instructions: 'Please fill in the required fields',
items: [
    {
    xtype: 'selectfield',
    name: 'type',
    label: 'Type',
    options: [{
                text: 'skoler(demo)', value: 'skoler'
                },{
                text: 'Fredet fortidsminde', value: 'fortidsminder'
                }
                ]
    },
    {
    xtype: 'sliderfield',
    label: 'Proximity(m)',
    minValue: 0,
    maxValue: 10000,
    increment: 50,
    name: 'radius',
    id: 'radius',
    plugins: [new Ext.ux.CustomSlider({
        valueUnit: ' m',
        tooltipStyle: 'background-color: #FFF; text-align: center',
        showSliderBothEndValue: false,
        sliderEndValuePos: 'center'
    })],
    }

],

},
{
    xtype: 'fieldset',
    title: 'Coordinates',
    instructions: 'Marker coordinates',
    items: [
        {
        xtype: 'textfield',
        name: 'lat',
        label: 'Latitude',
        useClearIcon: true,
        autoCapitalize: false,
        required: false,
        disabled: true
        }, {
        xtype: 'textfield',
        name: 'lng',
        label: 'Longitude',
        useClearIcon: true,
        autoCapitalize: false,
        required: false,
        disabled: true
```

```
                    }
                ],
            }],
        });


//---------------------------------------------------------------------------
//  Submitform - WITHIN
//---------------------------------------------------------------------------

    var submitWithinForm = new Ext.form.FormPanel({
            fullscreen: true,
            scroll: 'vertical',
            standardSubmit: false,
            id: "withinForm",
            title: 'withinForm',
            items: [{
                xtype: 'fieldset',
                title: 'User input',
                instructions: 'Please fill in the required fields',
                items: [
                    {
                    xtype: 'selectfield',
                    name: 'type',
                    label: 'Type',
                    options: [{
                                text: 'Skoler', value: 'skoler'
                                },
                                 {
                                text: 'Fortidsminder', value: 'fortidsminder'
                                },
                                {
                                text: 'Beskyttelses linjer', value: 'beskyt_linje'
                                }, {
                                text: 'Paragraf 3 omr.', value: 'paragraf3'
                                }]
                    },
                ],
                },
                {
                    xtype: 'fieldset',
                    title: 'Coordinates',
                    instructions: 'Coordinates array',
                    items: [
                        {
                        xtype: 'textfield',
                        name: 'array',
                        label: 'Array',
                        useClearIcon: true,
                        autoCapitalize: false,
                        required: false,
                        disabled: true
                        }
```

```
                    ],
            }],
        });



//---------------------------------------------------------------------------
//  Submitform - POLYGON
//---------------------------------------------------------------------------
    var submitPolygonForm = new Ext.form.FormPanel({
            fullscreen: true,
            scroll: 'vertical',
            standardSubmit: false,
            id: "polygonForm",
            title: 'polygonForm',
            items: [{
                xtype: 'fieldset',
                title: 'User input',
                instructions: 'Please fill in the required fields',
                items: [
                    {
                    xtype: 'textfield',
                    type: 'int',
                    name: 'name',
                    label: 'Name',
                    placeHolder: "enter your name",
                    useClearIcon: false,
                    autoCapitalize: false,
                    required: true,
                    },
                    {
                    xtype: 'datepickerfield',
                    slotOrder:
                                ['year', 'months', 'day']
                            ,
                    name: 'date',
                    label: 'Date',
                    useClearIcon: true,
                    autoCapitalize: false,
                    required: true
                    },
                    {
                    xtype: 'selectfield',
                    name: 'type',
                    label: 'Type',
                    options: [{
                                text: 'Mose', value: 'Beskyttede naturtyper : Mose'
                                }, {
                                text: 'Sø', value: 'Beskyttede naturtyper : Sø'
                                }, {
                                text: 'Overdrev', value: 'Beskyttede naturtyper :
```

```
Overdrev'
                                }, {
                                text: 'Strandeng', value: 'Beskyttede naturtyper :
Strandeng'
                                }, {
                                text: 'Eng', value: 'Beskyttede naturtyper : Eng'
                                }, {
                                text: 'Hede', value: 'Beskyttede naturtyper : Hede'
                                }]
                        },
                        {
                        xtype: 'textareafield',
                        name: 'comment',
                        label: 'Comments',
                        placeHolder: "Optional comment (300 characters or less)",
                        useClearIcon: false,
                        autoCapitalize: false,
                        required: false,
                        maxLength: 300
                        },
                    ],
                    },
                    {
                        xtype: 'fieldset',
                        title: 'Coordinates',
                        instructions: 'Coordinates array',
                        items: [
                            {
                            xtype: 'textfield',
                            name: 'array',
                            label: 'Array',
                            useClearIcon: true,
                            autoCapitalize: false,
                            required: false,
                            disabled: true
                            }

                        ],
                }],
            });

//---------------------------------------------------------------------------
//  Submitform - LINE
//---------------------------------------------------------------------------
    var submitLineForm = new Ext.form.FormPanel({
            fullscreen: true,
            scroll: 'vertical',
            standardSubmit: false,
            id: "lineForm",
            title: 'lineForm',
            items: [{
                xtype: 'fieldset',
                title: 'User input',
```

```
                instructions: 'Please fill in the required fields',
                items: [
                    {
                    xtype: 'textfield',
                    type: 'int',
                    name: 'name',
                    label: 'Name',
                    placeHolder: "enter your name",
                    useClearIcon: false,
                    autoCapitalize: false,
                    required: true,
                    },
                    {
                    xtype: 'datepickerfield',
                    slotOrder:
                                ['year', 'months', 'day']
                            ,
                    name: 'date',
                    label: 'Date',
                    useClearIcon: true,
                    autoCapitalize: false,
                    required: true
                    },
                    {
                    xtype: 'selectfield',
                    name: 'type',
                    label: 'Type',
                    options: [{
                                text: 'Beskyttede sten- og jorddiger', value:
'Beskyttede sten- og jorddiger'
                                }, {
                                text: 'Beskyttede vandløb', value: 'Beskyttede
vandløb'
                                }]
                    },
                    {
                    xtype: 'textareafield',
                    name: 'comment',
                    label: 'Comments',
                    placeHolder: "Optional comment (300 characters or less)",
                    useClearIcon: false,
                    autoCapitalize: false,
                    required: false,
                    maxLength: 300
                    },
                ],
                },
                {
                    xtype: 'fieldset',
                    title: 'Coordinates',
                    instructions: 'Coordinates array',
                    items: [
                        {
```

```
                        xtype: 'textfield',
                        name: 'array',
                        label: 'Array',
                        useClearIcon: true,
                        autoCapitalize: false,
                        required: false,
                        disabled: true
                        }
                    ],
            }],
        });


//-------------------------------------------------------------------------
//  Funktioner som bruges i forbindelse med beregning af længde og areal
//-------------------------------------------------------------------------

        function measurePoint(latLng) {
            // Add a draggable marker to the map where the user clicked
            var marker = new google.maps.Marker({
                    map: mapdemo.map,
                    position: latLng,
                    draggable: true,
                    raiseOnDrag: false,
                    title: "Drag me to change shape",
                    icon: new google.maps.MarkerImage("images/measure-vertex.png",
new google.maps.Size(9, 9), new google.maps.Point(0, 0), new google.maps.Point(5, 5))
                });
                    markersArray.push(marker);
        };

        function measureAdd(latLng) {
                // Add a draggable marker to the map where the user clicked
                var marker = new google.maps.Marker({
                    map: mapdemo.map,
                    position: latLng,
                    draggable: true,
                    raiseOnDrag: false,
                    title: "Drag me to change shape",
                    icon: new google.maps.MarkerImage("images/measure-vertex.png",
new google.maps.Size(9, 9), new google.maps.Point(0, 0), new google.maps.Point(5, 5))
                });

                // Add this LatLng to our line and polygon MVCArrays
                // Objects added to these MVCArrays automatically update the line
and polygon shapes on the map
                measure.mvcLine.push(latLng);
                measure.mvcPolygon.push(latLng);

                // Push this marker to an MVCArray
                // This way later we can loop through the array and remove them
when measuring is done
                measure.mvcMarkers.push(marker);
```

```
                // Get the index position of the LatLng we just pushed into the
MVCArray
                // We'll need this later to update the MVCArray if the user moves
the measure vertexes
                var latLngIndex = measure.mvcLine.getLength() - 1;

                // When the user mouses over the measure vertex markers, change
shape and color to make it obvious they can be moved
                google.maps.event.addListener(marker, "mouseover", function() {
                    marker.setIcon(new google.maps.MarkerImage(
"images/measure-vertex-hover.png", new google.maps.Size(20, 20), new google.maps.
Point(0, 0), new google.maps.Point(10, 10)));
                });

                // Change back to the default marker when the user mouses out
                google.maps.event.addListener(marker, "mouseout", function() {
                    marker.setIcon(new google.maps.MarkerImage(
"images/measure-vertex.png", new google.maps.Size(9, 9), new google.maps.Point(0, 0
), new google.maps.Point(5, 5)));
                });

                // When the measure vertex markers are dragged, update the geometry
of the line and polygon by resetting the
                //     LatLng at this position
                google.maps.event.addListener(marker, "drag", function(evt) {
                    measure.mvcLine.setAt(latLngIndex, evt.latLng);
                    measure.mvcPolygon.setAt(latLngIndex, evt.latLng);
                });

                // When dragging has ended and there is more than one vertex,
measure length, area.
                google.maps.event.addListener(marker, "dragend", function() {
                    if (measure.mvcLine.getLength() > 1) {
                        measureCalc();
                    }
                });

                // If there is more than one vertex on the line
                if (measure.mvcLine.getLength() > 1) {

                    // If the line hasn't been created yet
                    if (!measure.line) {

                        // Create the line (google.maps.Polyline)
                        measure.line = new google.maps.Polyline({
                            map: mapdemo.map,
                            clickable: false,
                            strokeColor: "#6069EB",
                            strokeOpacity: 1,
                            strokeWeight: 3,
                            path:measure. mvcLine
                        });
```

```
                    }

                    // If there is more than two vertexes for a polygon
                    if (measure.mvcPolygon.getLength() > 2) {

                        // If the polygon hasn't been created yet
                        if (!measure.polygon) {

                            // Create the polygon (google.maps.Polygon)
                            measure.polygon = new google.maps.Polygon({
                                clickable: false,
                                map: mapdemo.map,
                                fillColor: "#6069EB",
                                fillOpacity: opacity,
                                strokeOpacity: 0,
                                paths: measure.mvcPolygon
                            });

                        }

                    }

                }

                // If there's more than one vertex, measure length, area.
                if (measure.mvcLine.getLength() > 1) {
                    measureCalc();
                }

            }

        function measureCalc() {

            // Use the Google Maps geometry library to measure the length of the line
            length = google.maps.geometry.spherical.computeLength(measure.line.
getPath());
            jQuery("#span-length").text(length.toFixed(1));
            jQuery("#span-length2").text(length.toFixed(1));
            //console.log(length);

            // If we have a polygon (>2 vertexes inthe mvcPolygon MVCArray)
            if (measure.mvcPolygon.getLength() > 2) {
                // Use the Google Maps geometry library to measure the area of the
polygon
                area = google.maps.geometry.spherical.computeArea(measure.polygon.
getPath());
                jQuery("#span-area").text(area.toFixed(1));
            //  console.log(area);
            }

        }
```

```
        var clearPolygon = function measureReset() {

        // If we have a polygon or a line, remove them from the map and set null
        if (measure.polygon) {
            measure.polygon.setMap(null);
            measure.polygon = null;
        }
        if (measure.line) {
            measure.line.setMap(null);
            measure.line = null
        }

        // Empty the mvcLine and mvcPolygon MVCArrays
        measure.mvcLine.clear();
        measure.mvcPolygon.clear();

        // Loop through the markers MVCArray and remove each from the map, then
empty it
        measure.mvcMarkers.forEach(function(elem, index) {
            elem.setMap(null);
        });
        measure.mvcMarkers.clear();


    }

//-------------------------------------------------------------------------
//  KORT - styrring af basis funktioner
//-------------------------------------------------------------------------

    //------------- Definerer korttype for første kortvisning -----------
    var typex = google.maps.MapTypeId.ROAD  //korttype: google.roadmap

    //------------- Følgende kører i Google maps API v3 -----------------
    var mapdemo = new Ext.Map({ //
            title: 'map',
            useCurrentLocation: false,
            mapOptions : {
                center : new google.maps.LatLng(positionArr[0], positionArr[1]),
// centrerer på seneste sted kortet har været (hvis ikke angivet = København)
                maxZoom : 21,
                minZoom: 1,
                zoom : positionArr[2],      // seneste zoomniveau - hvis ikke brugt
før = 12
                mapTypeId : positionArr[3],     // ved opstart Roadmap, ellers det
af brugeren valgte
                mapTypeControl: false,
                navigationControl: (Ext.is.Andriod? true : false),
            },


        listeners : {       // skriver kortcenter, zoom og type i lokal hukommelse
når kortet ændres.
```

```
                maprender: function(extmap, map){
                    ExtMap.MapTypes.registerMaps(map);

                    google.maps.event.addListener(map, 'maptypeid_changed', function
() {
                        ExtMap.Storage.saveLastPosition(map)
                    });
                    google.maps.event.addListener(map, 'center_changed', function() {
                        ExtMap.Storage.saveLastPosition(map)
                    });

                    google.maps.event.addListener(map, 'zoom_changed', function() {
                        ExtMap.Storage.saveLastPosition(map)
                    });


            var KMLLayer = new google.maps.KmlLayer(
'http://ukiyo.dk/KMLpoints23jan.kml', {preserveViewport:true});
            var KMLLayer = new google.maps.KmlLayer(
'http://2.107.248.183:8080/geoserver/wms/reflect?format=kml&layers=temp', {
preserveViewport:true, supressInfoWindows: true});
            KMLLayer.setMap(mapdemo.map);


            var POLYLayer = new google.maps.KmlLayer(
'http://ukiyo.dk/Peter-polygons1.kml', {preserveViewport:true});
            POLYLayer.setMap(mapdemo.map);


        var imageMapTypeOptions = { //Overlay til baggrundskortet hentet fra
Geoserver via GeoWebCache
                        getTileUrl: function(coord, zoom) {
                            return
'http://2.107.248.183:8080/geoserver/gwc/service/gmaps?layers=geoApp&zoom=' + zoom +
 '&x=' + coord.x + '&y=' + coord.y + '&format=image/png';
                        },
                            tileSize: new google.maps.Size(256, 256),
                            isPng: true,
                            opacity: 0.5
                        }

        var tiledImageMap = new google.maps.ImageMapType(imageMapTypeOptions);

                    map.overlayMapTypes.insertAt(0, tiledImageMap);
                }
            }

    });



//-------------------------------------------------------------------------
//  SETUP
```

```
//-----------------------------------------------------------------------------
Ext.setup({
    tabletStartupScreen: 'img/tablet_startup.png',
    phoneStartupScreen: 'img/phone_startup.png',
    icon: 'icon.png',
    glossOnIcon: false,
    onReady: function() {

    Ext.apply(Ext.util.Format, {
    defaultDateFormat: 'Y-m-d'
});

//-----------------------------------------------------------------------------
//  TOOLBAR - placeret i bunden af skærmen indeholdende applikationens knapper
//-----------------------------------------------------------------------------
        var toolbar = new Ext.TabBar({
            dock : 'bottom',
            id: 'toolbarId',
            ui   : 'light',
            hidden: false,
            layout: {pack: 'center'},
            items: [
                {
                    iconMask: true,
                    text: 'Locate',
                    title: 'locate',
                    id: 'locateid',
                    iconCls: 'locate',
                    handler: ExtMap.UI.showLocateListOverlay,
                    //ExtMap.Utils.geolocation,

                    },
                    {
                    iconMask: true,
                    text: 'Tools',
                    title: 'delete',
                    id: 'toolid',
                    iconCls: 'settings3',
                    handler: ExtMap.UI.showToolListOverlay,
                    },
                    {
                    iconMask: true,
                    text: 'Cancel',
                    id: 'cancelid',
                    title: 'cancel',
                    iconCls: 'delete',
                    hidden: true,
                    handler: cancelHandler,
                    },
                    {
                    iconMask: true,
                    text: 'Add',
                    title: 'add',
```

```
id: 'addid',
iconCls: 'compose',
handler: ExtMap.UI.showAddListOverlay,
//addHandler,
},
{
iconMask: true,
text: 'Choose map',
title: 'map chooser',
id: 'choosemapid',
iconCls: 'maps',
handler: ExtMap.UI.showMapListOverlay,
},
{
iconMask: true,
text: 'Legend',
title: 'signatur',
id: 'signaturid',
iconCls: 'list' ,
handler: tapHandler,
},
{
xtype: 'spacer',
id: 'spacerid',
hidden: true
},
{
iconMask: true,
text: 'submit',
id: 'submitid',
title: 'submit',
iconCls: 'check_black2',
hidden: true,
handler: submitCardHandler,
},
{
iconMask: true,
text: 'submit',
id: 'submitPolyId',
title: 'submit',
iconCls: 'check_black2',
hidden: true,
handler: submitPolyCardHandler,
},
{
iconMask: true,
text: 'submit',
id: 'submitProxId',
title: 'submit',
iconCls: 'check_black2',
hidden: true,
handler: submitProxCardHandler,
},
```

```
                    {
                    iconMask: true,
                    text: 'submit',
                    id: 'submitWithinId',
                    title: 'submit',
                    iconCls: 'check_black2',
                    hidden: true,
                    handler: submitWithinCardHandler,
                    },
                    {
                    iconMask: true,
                    text: 'submit',
                    id: 'submitLineId',
                    title: 'submit',
                    iconCls: 'check_black2',
                    hidden: true,
                    handler: submitLineCardHandler,
                    }
            ]

        });

//-----------------------------------------------------------------------------
//  mapPanel - indeholdende baggrundskort
//-----------------------------------------------------------------------------
    mapPanel = new Ext.Panel({
            dockedItems: [toolbar],
            fullscreen: true,
            layout: 'card',
            items: [mapdemo]
        });




//-----------------------------------------------------------------------------
//  proxListPanel - indeholdende baggrundskort
//-----------------------------------------------------------------------------


     proxList= new Ext.List({
            id: 'proxList',
            store: searchStore,
            itemTpl:'{navn} (fot id: {fot_id}) {objekttkst}',
             onItemDisclosure: function(record, btn, index) {
                //proxPanel.toolbar.setTitle('{navn}');
                console.log(record);
                proxPanel.update(record);
                Panel.setActiveItem('proxPanel');
            }
        });

    proxListPanel = new Ext.Panel({
```

```
            id: 'proxListPanel',
            items:[proxList],
            dockedItems: [{
                xtype: 'toolbar',
                ui: 'light',
                title: 'Nearest schools',
                items: [{
                    text: 'Back',
                    ui: 'back',
                    handler: backHandler
                }]
            }],
            fullscreen: true,
            layout: 'card',
        });


//---------------------------------------------------------------------------
//  proxPanel - indeholdende baggrundskort
//---------------------------------------------------------------------------
    proxPanel = new Ext.Panel({
            id: 'proxPanel',
            tpl: '<b>{data.navn}</b> <br /><br /><u>Adresse:</u><br
/>{data.vejnavn} {data.vejnummer}<br />{data.postnummer} {data.bby}<br /><br
/><u>Kontakt:</u><br />Tel: {data.telefonnum} <br /> Email: <a
href="mailto:{data.mailadress}">{data.mailadress} </a><br />Hjemmeside: <a
href="{data.hjemmeside}" target="_blank">{data.hjemmeside}</a><br /><br /><img
src="http://maps.googleapis.com/maps/api/staticmap?markers={data.vejnavn}+denmark+{da
ta.vejnummer}+copenhagen+{data.postnummer}&zoom=14&size=400x350&sensor=false">',
            dockedItems: [{
                xtype: 'toolbar',
                ui: 'light',
                title: 'School info',
                items: [{
                    text: 'Back',
                    ui: 'back',
                    handler: backToListHandler
                }]
            }],
            fullscreen: true,
            scroll: 'vertical',
            layout: 'card',
        });

//---------------------------------------------------------------------------
//  infoBar - visnin af længde og areal ved tilføj/mål funktionerne
//---------------------------------------------------------------------------
    var infoBar = new Ext.Toolbar({
            dock : 'top',
            id: 'infoBarId',
            ui: 'light',
            showAnimation: "fade",
            style: {
```

```
                    width: '100%',
                    height: '25px',
                },
            html: "<div style='color:#FFF'><p>Length: <span
id='span-length'>0</span> m - Area: <span id='span-area'>0</span> m²</p></div>",
            hidden: true,
        });


//-------------------------------------------------------------------------------
//  infoBar1 - visnin af længde og areal ved tilføj/mål funktionerne
//-------------------------------------------------------------------------------
    var infoBar1 = new Ext.Toolbar({
            dock : 'top',
            id: 'infoBar1Id',
            ui: 'light',
            showAnimation: "fade",
            style: {
                    width: '100%',
                    height: '25px',
                },
            html: "<div style='color:#FFF'><p>Length: <span
id='span-length2'>0</span> m </p></div>",
            hidden: true,
        });


//-------------------------------------------------------------------------------
//  SIGNATURFORKLARING - panel indeholdende signatur_list
//-------------------------------------------------------------------------------
    signaturCard = new Ext.Panel({
            layout: 'card',
            dockedItems: [{
                xtype: 'toolbar',
                ui: 'light',
                title: 'Map Legend',
                items: [{
                    text: 'Back',
                    ui: 'back',
                    handler: backHandler
                }]
            }],
            items:[signatur_list]
        });

//-------------------------------------------------------------------------------
//  SUBMITCARD - panel submitformen til punkt
//-------------------------------------------------------------------------------
    submitCard = new Ext.Panel({
            layout: 'card',
                dockedItems: [{
                    xtype: 'tabbar',
                    dock: 'bottom',
```

```
                    ui: 'light',
                    title: 'submitcard',
                    items: [{
                        text: 'cancel',
                        title: 'cancel',
                        iconCls: 'delete',
                        handler: backHandler
                            },
                            {
                            xtype: 'spacer'
                            },
                            {
                        iconMask: true,
                        text: 'submit',
                        title: 'submit',
                        iconCls: 'check_black2',
                        handler: submitHandler,
                            }
                            ]
                        }],
                    items:[submitForm]
                });

//---------------------------------------------------------------------------
//  SUBMITCARD - panel submitformen til polygon
//---------------------------------------------------------------------------
    submitPolygonCard = new Ext.Panel({
            layout: 'card',
                dockedItems: [{
                    xtype: 'tabbar',
                    dock: 'bottom',
                    ui: 'light',
                    title: 'submitPolygoncard',
                    items: [{
                        text: 'cancel',
                        title: 'cancel',
                        iconCls: 'delete',
                        handler: backHandler
                            },
                            {
                            xtype: 'spacer'
                            },
                            {
                        iconMask: true,
                        text: 'submit',
                        title: 'submit',
                        iconCls: 'check_black2',
                        handler: submitPolygonHandler,
                            }
                            ]
                        }],
                    items:[submitPolygonForm]
                });
```

```
//-------------------------------------------------------------------------------
//  SUBMITCARD - panel submitformen til within
//-------------------------------------------------------------------------------
    submitWithinCard = new Ext.Panel({
            layout: 'card',
                dockedItems: [{
                    xtype: 'tabbar',
                    dock: 'bottom',
                    ui: 'light',
                    title: 'submitWithinCard',
                    items: [{
                        text: 'cancel',
                        title: 'cancel',
                        iconCls: 'delete',
                        handler: backHandler
                            },
                            {
                            xtype: 'spacer'
                            },
                            {
                        iconMask: true,
                        text: 'submit',
                        title: 'submit',
                        iconCls: 'check_black2',
                        handler: submitWithinHandler,
                            }
                            ]
                        }],
                    items:[submitWithinForm]
                });



//-------------------------------------------------------------------------------
//  SUBMITCARD - panel submitformen til line
//-------------------------------------------------------------------------------
    submitLineCard = new Ext.Panel({
            layout: 'card',
                dockedItems: [{
                    xtype: 'tabbar',
                    dock: 'bottom',
                    ui: 'light',
                    title: 'submitLinecard',
                    items: [{
                        text: 'cancel',
                        id: 'submitLineCardCancelId',
                        title: 'cancel',
                        iconCls: 'delete',
                        handler: backHandler
                            },
                            {
                            xtype: 'spacer'
```

```
                        },
                        {
                    iconMask: true,
                    text: 'submit',
                    id: 'submitLineCardId',
                    title: 'submit',
                    iconCls: 'check_black2',
                    handler: submitLineHandler,
                        }
                        ]
                    }],
                items:[submitLineForm]
            });

//---------------------------------------------------------------------------
//  SUBMITPROXCARD - panel proximity card
//---------------------------------------------------------------------------
    submitProxCard = new Ext.Panel({
            layout: 'card',
                dockedItems: [{
                    xtype: 'tabbar',
                    dock: 'bottom',
                    ui: 'light',
                    title: 'submitProxCard',
                    items: [{
                        text: 'cancel',
                        title: 'cancel',
                        iconCls: 'delete',
                        handler: backHandler
                        },
                        {
                        xtype: 'spacer'
                        },
                        {
                    iconMask: true,
                    text: 'submit',
                    title: 'submit',
                    iconCls: 'check_black2',
                    handler: submitProxHandler,
                        }
                        ]
                    }],
                items:[submitProxForm]
            });

//---------------------------------------------------------------------------
//  PANEL - "moderpanelet"
//---------------------------------------------------------------------------
    Panel = new Ext.Panel({
            dockedItems: [infoBar, infoBar1,  ],
            listeners: {
                el: { //el står for 'element' og er det DOM element der indeholder
vores panel. der er også en "body" i DOM elementet man kunne have benyttet
```

```
                    tap: function() {console.log('tab on body');},
                    single: true // single sat til true gør at denne listner
function kun kører én gang.
                    },
                },
            fullscreen: true,
            layout: 'card',
            cardSwitchAnimation: 'slide',
            items: [mapPanel, signaturCard, submitCard, submitWithinCard, proxPanel,
 proxListPanel, ]

        });
    }
});
```

```
ExtMap = {};
ExtMap.Utils = {};
ExtMap.UI = {};
ExtMap.Storage = {};
ExtMap.MapTypes = {};
ExtMap.Map = {};
ExtMap.Map.elevationSrv = new google.maps.ElevationService();
ExtMap.Tool= {};
ExtMap.Add= {};
ExtMap.Locate= {};


ExtMap.Map.geocoder = new google.maps.Geocoder();


 var listnode;
 var search;



//  =============   FUNKTION TIL VALG AF FORSKELLIGE BAGGRUNDSKORT  =============
// omregning til brug ifm. Bing-maps
ExtMap.Utils.TileToQuadKey = function (x, y, zoom) {
    var quad = "";
    for (var i = zoom; i > 0; i--) {
        var mask = 1 << (i - 1);
        var cell = 0;
        if ((x & mask) != 0) cell++;
        if ((y & mask) != 0) cell += 2;
        quad += cell
    }
    return quad
};

// definerer  variable til Bing, OSM og blankt kort.
ExtMap.MapTypes.BING_MAPS_ROAD = 'bingroad';
ExtMap.MapTypes.BING_MAPS_HYBRID = 'binghybrid';
ExtMap.MapTypes.BING_MAPS_SATELLITE = 'bingsatellite';
ExtMap.MapTypes.OSM_MAPS_ROAD = 'osmroad';
ExtMap.MapTypes.BLANK = 'blankmap';

//--- Laver liste med de forskellige baggrundskort (titel + link til tilhørende
billede) ---
    ExtMap.MapTypes.mapListData = [{
    mapTypeName: 'Google - Road',
    imgurl: 'img/google.png',
    typex: google.maps.MapTypeId.ROADMAP
}, {
    mapTypeName: 'Google - Satellite',
    imgurl: 'img/google.png',
    typex: google.maps.MapTypeId.SATELLITE
}, {
    mapTypeName: 'Google - Hybrid',
```

```
    imgurl: 'img/google.png',
    typex: google.maps.MapTypeId.HYBRID
}, {
    mapTypeName: 'Google - Terrain',
    imgurl: 'img/google.png',
    typex: google.maps.MapTypeId.TERRAIN
}, {
    mapTypeName: 'Bing - Road',
    imgurl: 'img/Bing.png',
    typex: ExtMap.MapTypes.BING_MAPS_ROAD
}, {
    mapTypeName: 'Bing - Satellite',
    imgurl: 'img/Bing.png',
    typex: ExtMap.MapTypes.BING_MAPS_SATELLITE
}, {
    mapTypeName: 'Bing - Hybrid',
    imgurl: 'img/Bing.png',
    typex: ExtMap.MapTypes.BING_MAPS_HYBRID
}, {
    mapTypeName: 'Open Street Maps',
    imgurl: 'img/OSM.png',
    typex: ExtMap.MapTypes.OSM_MAPS_ROAD
}, {
    mapTypeName: 'Blank Map',
    imgurl: 'img/blank.png',
    typex: ExtMap.MapTypes.BLANK
}
];
//--- Laver liste med de forskellige Tools (marker punkt, linje og mål) ---
    ExtMap.Tool.toolListData = [{
    toolName: 'Near Point',
    imgurl: 'img/bullseye1.png',
    toolx: 1,
}, {
    toolName: 'Near Line',
    imgurl: 'img/linebuffer.png',
    toolx: 2,
},
{
    toolName: 'Within Polygon',
    imgurl: 'img/polygonbuffer.png',
    toolx: 3,
},
 {
    toolName: 'Measure length',
    imgurl: 'img/ruller.png',
    toolx:4,
},
 {
    toolName: 'Measure area',
    imgurl: 'img/ruller2.png',
    toolx:5,
},
```

```
 {
    toolName: 'Get Coordinates',
    imgurl: 'img/XY.png',
    toolx:6,
}

];
//--------------------SLUT list Tools ----------------

//--- Laver liste med de forskellige Add (marker punkt, linje og mål) ---
    ExtMap.Add.addListData = [{
    addName: 'Point',
    imgurl: 'img/locate.png',
    addx: 1,
},
{
    addName: 'Line',
    imgurl: 'img/draw_line.png',
    addx: 2,
},
{
    addName: 'Polygon',
    imgurl: 'img/draw_polygon.png',
    addx: 3,
}
];
//--- SLUT list Add -----------

//--- Laver liste med de forskellige Locate (marker punkt, linje og mål) ---
    ExtMap.Locate.locateListData = [{
    locateName: 'Current location',
    imgurl: 'img/locate.png',
    locatex: 1,
},
{
    locateName: 'Find address',
    imgurl: 'img/search1.png',
    locatex: 2,
}
];
//--- SLUT list Locate ---


// registrerer Bing maps, OSM og blankt baggrundskort
ExtMap.MapTypes.registerMaps = function (map) {
    var bingRoadMapType = new ExtMap.MapTypes.bingMapsRoadLayer();
    map.mapTypes.set(ExtMap.MapTypes.BING_MAPS_ROAD, bingRoadMapType);
    var bingHybridMapType = new ExtMap.MapTypes.bingMapsHybridLayer();
    map.mapTypes.set(ExtMap.MapTypes.BING_MAPS_HYBRID, bingHybridMapType);
    var bingSatelliteMapType = new ExtMap.MapTypes.bingMapsSatelliteLayer();
    map.mapTypes.set(ExtMap.MapTypes.BING_MAPS_SATELLITE, bingSatelliteMapType);
    var OSMRoadMapType = new ExtMap.MapTypes.OSMRoadLayer();
    map.mapTypes.set(ExtMap.MapTypes.OSM_MAPS_ROAD, OSMRoadMapType);
```

```
    var blankMapType = new ExtMap.MapTypes.blankMapType();
    map.mapTypes.set(ExtMap.MapTypes.BLANK, blankMapType)
};

// design af kald til Bing
ExtMap.MapTypes.getMapTilesQuad = function (coord, zoom, ownerDocument, url1, url2,
url3) {
    var div = ownerDocument.createElement('DIV');
    var tempQuad = ExtMap.Utils.TileToQuadKey(coord.x, coord.y, zoom);
    var quadTile = '000000';
    quadTile += (parseInt(coord.y.toString(2) * 2) + parseInt(coord.x.toString(2)));
    quadTile = quadTile.substring(quadTile.length - zoom, quadTile.length);
    tileUrl = url1 + quadTile.substring(quadTile.length - 1, quadTile.length) + url2
 + tempQuad + url3;
    div.innerHTML = '<img src="' + tileUrl + '">';
    return div
};

// design af kald til OSM
ExtMap.MapTypes.getMapTilesOSM = function (coord, zoom, ownerDocument) {
    var div = ownerDocument.createElement('DIV');
    var servers = new Array("a", "a", "b", "c");
    var rand_no = Math.ceil(Math.random() * 3);
    tileUrl = 'http://' + servers[rand_no] + '.tile.openstreetmap.org/' + zoom + '/'
 + coord.x + '/' + coord.y + '.png';
    div.innerHTML = '<img src="' + tileUrl + '">';
    return div
};

// Kald til Bing - road map
ExtMap.MapTypes.bingMapsRoadLayer = function () {};
ExtMap.MapTypes.bingMapsRoadLayer.prototype.tileSize = new google.maps.Size(256, 256
);
ExtMap.MapTypes.bingMapsRoadLayer.prototype.maxZoom = 19;
ExtMap.MapTypes.bingMapsRoadLayer.prototype.getTile = function (coord, zoom,
ownerDocument) {
    return ExtMap.MapTypes.getMapTilesQuad(coord, zoom, ownerDocument,
'http://ecn.t', '.tiles.virtualearth.net/tiles/r',
'.png?g=563&mkt=en-us&lbl=l1&stl=h&shading=hill&n=z')
    };
ExtMap.MapTypes.bingMapsRoadLayer.prototype.name = "Bing Maps";
ExtMap.MapTypes.bingMapsRoadLayer.prototype.alt = "Bing Maps Road";


// Kald til Bing - hybrid map
ExtMap.MapTypes.bingMapsHybridLayer = function () {};
ExtMap.MapTypes.bingMapsHybridLayer.prototype.tileSize = new google.maps.Size(256,
256);
ExtMap.MapTypes.bingMapsHybridLayer.prototype.maxZoom = 19;
ExtMap.MapTypes.bingMapsHybridLayer.prototype.getTile = function (coord, zoom,
ownerDocument) {
    return ExtMap.MapTypes.getMapTilesQuad(coord, zoom, ownerDocument,
'http://ecn.t', '.tiles.virtualearth.net/tiles/h', '.jpeg?g=563&mkt=en-us&n=z')
```

```
    };
ExtMap.MapTypes.bingMapsHybridLayer.prototype.name = "Bing Maps";
ExtMap.MapTypes.bingMapsHybridLayer.prototype.alt = "Bing Maps Hybrid";


// Kald til Bing - satellite map
ExtMap.MapTypes.bingMapsSatelliteLayer = function () {};
ExtMap.MapTypes.bingMapsSatelliteLayer.prototype.tileSize = new google.maps.Size(256
, 256);
ExtMap.MapTypes.bingMapsSatelliteLayer.prototype.maxZoom = 19;
ExtMap.MapTypes.bingMapsSatelliteLayer.prototype.getTile = function (coord, zoom,
ownerDocument) {
    return ExtMap.MapTypes.getMapTilesQuad(coord, zoom, ownerDocument,
'http://ecn.t', '.tiles.virtualearth.net/tiles/a', '.jpeg?g=563&mkt=en-us&n=z')
    };
ExtMap.MapTypes.bingMapsSatelliteLayer.prototype.name = "Bing Maps";
ExtMap.MapTypes.bingMapsSatelliteLayer.prototype.alt = "Bing Maps Satellite";


// Kald til OSM
ExtMap.MapTypes.OSMRoadLayer = function () {};
ExtMap.MapTypes.OSMRoadLayer.prototype.tileSize = new google.maps.Size(256, 256);
ExtMap.MapTypes.OSMRoadLayer.prototype.maxZoom = 19;
ExtMap.MapTypes.OSMRoadLayer.prototype.getTile = function (coord, zoom,
ownerDocument) {
    return ExtMap.MapTypes.getMapTilesOSM(coord, zoom, ownerDocument)
    };
ExtMap.MapTypes.OSMRoadLayer.prototype.name = "Open Street Maps";
ExtMap.MapTypes.OSMRoadLayer.prototype.alt = "Open Street Maps Road";

// Kald til blankt baggrundskort
ExtMap.MapTypes.blankMapType = function () {};
ExtMap.MapTypes.blankMapType.prototype.tileSize = new google.maps.Size(256, 256);
ExtMap.MapTypes.blankMapType.prototype.maxZoom = 19;
ExtMap.MapTypes.blankMapType.prototype.getTile = function (coord, zoom,
ownerDocument) {
    var div = ownerDocument.createElement('DIV');
    div.style.width = this.tileSize.width + 'px';
    div.style.height = this.tileSize.height + 'px';
    div.style.borderColor = '#AAAAAA';
    return div
};
ExtMap.MapTypes.blankMapType.prototype.name = "Blank Map";
ExtMap.MapTypes.blankMapType.prototype.alt = "Blank Map Type";


//########################################


Ext.regModel('searchList', {
    fields: [
            {name: 'gid', type: 'int'},
            {name: 'skoleid', type: 'int'},
```

```
            {name: 'fot_id', type: 'int'},
            {name: 'objekttkst', type: 'string'},
            {name: 'y_koordina', type: 'int'},
            {name: 'x_koordina', type: 'int'},
            {name: 'vejkode', type: 'int'},
            {name: 'kommunekod', type: 'int'},
            {name: 'omraade', type: 'string'},
            {name: 'navn', type: 'string'},
            {name: 'vejnavn', type: 'string'},
            {name: 'vejnummer', type: 'string'},
            {name: 'postnummer', type: 'int'},
            {name: 'bby', type: 'string'},
            {name: 'telefonnum', type: 'string'},
            {name: 'mailadress', type: 'string'},
            {name: 'hjemmeside', type: 'string'},
            {name: 'objectid', type: 'int'},
            {name: 'id1', type: 'int'},
            {name: 'skolevejsp', type: 'string'},
            {name: 'gmrotation', type: 'int'},
            {name: 'orig_fid', type: 'int'},
            {name: 'the_geom', type: 'string'}
            ],
    //idProperty: 'gid',
    });


            searchStore = new Ext.data.Store({
                id: 'searchStore',
                autoLoad: false,
                model: 'searchList',
                proxy: {
                    type: 'ajax',
                    url: '',
                    reader: {
                        type: 'json',
                        root: 'search'
                    }
                }
            });


        //submitProxHandler.getArray().search
//################################################


//-------------------------------------------------------------------------------
------------
//   Laver en "ListStore" indeholdende elementerne fra map listen defineret i
toppen af koden
//-------------------------------------------------------------------------------
------------
```

```
    Ext.regModel('mapTypeList', {
        fields: ['mapTypeName', 'imgurl']
    });

    ExtMap.UI.mapListStore = new Ext.data.Store({
    model: 'mapTypeList',
    getGroupString: function (record) {
        return record.get('mapTypeName')[0]
    },
    data: ExtMap.MapTypes.mapListData
    });

    ExtMap.UI.mapTypeList = new Ext.List({
    width: Ext.is.Phone ? undefined : 220,
    height: 195,
    store: ExtMap.UI.mapListStore,

    itemTpl: '<tpl for="."><div
class="maptypelist"><strong>{mapTypeName}</strong><img src=\'{imgurl}\'
align=\'right\'></div></tpl>',
    itemSelector: 'div.maptypelist',
    singleSelect: true,
    grouped: false,
    indexBar: false
    });

    ExtMap.UI.mapTypeList.on('itemtap', function (list, index, item, e) {
    //mapdemo.map.setMapTypeId(ExtMap.MapTypes.BLANK);
    mapPanel.setActiveItem(new Ext.Panel({
        items: [mapdemo]
    }), {
        type: 'fade',
        duration: 1000
    });
    mapdemo.map.setMapTypeId(list.store.getAt(index).get('typex'));
    ExtMap.UI.mapListOverlay.hide()
    });

    ExtMap.UI.overlayTb = new Ext.Toolbar({
    dock: 'top',
    });

    ExtMap.UI.mapListOverlay = new Ext.Panel({
    floating: true,
    modal: true,
    centered: false,
    bodyPadding: 3,
    bodyMargin: 3,
    width: Ext.is.Phone ? 240 : 240,
    height: Ext.is.Phone ? 220 : 220,
    //dockedItems: ExtMap.UI.overlayTb,
    scroll: 'vertical',
```

```
    scrollbar: 'show',
    items: [ExtMap.UI.mapTypeList]
    });

    ExtMap.UI.showMapListOverlay = function (btn, event) {
        ExtMap.UI.mapListOverlay.setCentered(false);
        //ExtMap.UI.overlayTb.setTitle('Base map');
        ExtMap.UI.mapListOverlay.showBy(btn)
    };

//-----------------------------------------------------------------------------
--------------
//                                  MapList Slut
//-----------------------------------------------------------------------------
--------------



//-----------------------------------------------------------------------------
------------
//   Laver en "ListStore" indeholdende elementerne fra Add-listen defineret i
toppen af koden
//-----------------------------------------------------------------------------
------------

Ext.regModel('addList', {
        fields: ['addName', 'imgurl']
    });

    ExtMap.UI.addListStore = new Ext.data.Store({
    model: 'addList',
    getGroupString: function (record) {
        return record.get('addName')[0]
    },
    data: ExtMap.Add.addListData
    });

    ExtMap.UI.addList = new Ext.List({
    width: Ext.is.Phone ? undefined : 200,
    height: 140,
    store: ExtMap.UI.addListStore,
    itemTpl: '<tpl for="."><div class="addtypelist"><strong>{addName}</strong><img
src=\'{imgurl}\' align=\'right\'></div></tpl>',
    itemSelector: 'div.addtypelist',
    singleSelect: true,
    grouped: false,
    indexBar: false,
    scroll: false,
    });


    ExtMap.UI.addList.on('itemtap', function (list, index, item, e) {
        listnode = list.store.getAt(index).get('addx');
        console.log(list.store.getAt(index).get('addx'));
```

```
        // Hvis "add point":
        if  (listnode == 1) {
            Ext.getCmp('locateid').hide();
            Ext.getCmp('toolid').hide();
            Ext.getCmp('submitPolyId').hide();
            Ext.getCmp('signaturid').hide();
            Ext.getCmp('addid').hide();
            Ext.getCmp('submitid').show();
            Ext.getCmp('spacerid').show();
            Ext.getCmp('cancelid').show();
            Ext.getCmp('submitProxId').hide();
            google.maps.event.addListenerOnce(mapdemo.map, "mousedown", function(evt
) {
                        measurePoint(evt.latLng);
                        markerLat = evt.latLng.lat();
                        markerLng = evt.latLng.lng();
                    });
            }

        // Hvis "add line":
        if  (listnode == 2) {
            opacity = 0,
            Ext.getCmp('locateid').hide();
            Ext.getCmp('toolid').hide();
            Ext.getCmp('signaturid').hide();
            Ext.getCmp('addid').hide();
            Ext.getCmp('submitid').hide();
            Ext.getCmp('submitLineId').show();
            Ext.getCmp('spacerid').show();
            Ext.getCmp('cancelid').show();
            Ext.getCmp('infoBar1Id').show();
            Ext.getCmp('submitProxId').hide();
            google.maps.event.addListener(mapdemo.map, "mousedown", function(evt) {
                        measureAdd(evt.latLng);
                        markerLat = evt.latLng.lat();
                        markerLng = evt.latLng.lng();
                    });
            }

        // Hvis "Add Polygon":
        if  (listnode == 3) {
            opacity = 0.25,
            Ext.getCmp('locateid').hide();
            Ext.getCmp('toolid').hide();
            Ext.getCmp('signaturid').hide();
            Ext.getCmp('addid').hide();
            Ext.getCmp('submitid').hide();
            Ext.getCmp('submitPolyId').show();
            Ext.getCmp('spacerid').show();
            Ext.getCmp('cancelid').show();
            Ext.getCmp('infoBarId').show();
```

```
            Ext.getCmp('submitProxId').hide();
            google.maps.event.addListener(mapdemo.map, "mousedown", function(evt) {
                        measureAdd(evt.latLng);
                        markerLat = evt.latLng.lat();
                        markerLng = evt.latLng.lng();
                    });
            }


                ExtMap.UI.addListOverlay.hide()
    });


    ExtMap.UI.addListOverlay = new Ext.Panel({
    floating: true,
    modal: true,
    centered: false,
    bodyPadding: 3,
    bodyMargin: 3,
    width: Ext.is.Phone ? 222 : 222,
    height: Ext.is.Phone ? 170 : 170,
    scroll: false,
    items: [ExtMap.UI.addList]
    });

    ExtMap.UI.showAddListOverlay = function (btn, event) {
        ExtMap.UI.addListOverlay.setCentered(false);
        ExtMap.UI.addListOverlay.showBy(btn)
    };

//-----------------------------------------------------------------------------
------------
//                                  AddList Slut
//-----------------------------------------------------------------------------
------------

//-----------------------------------------------------------------------------
------------
//   Laver en "ListStore" indeholdende elementerne fra Tool-listen defineret i
toppen af koden
//-----------------------------------------------------------------------------
------------
    Ext.regModel('toolList', {
        fields: ['toolName', 'imgurl']
    });

    ExtMap.UI.toolListStore = new Ext.data.Store({
    model: 'toolList',
    getGroupString: function (record) {
        return record.get('toolName')[0]
    },
    data: ExtMap.Tool.toolListData
    });
```

```
    ExtMap.UI.toolList = new Ext.List({
    width: Ext.is.Phone ? undefined : 200,
    height: 195,
    store: ExtMap.UI.toolListStore,
    itemTpl: '<tpl for="."><div
class="tooltypelist"><strong>{toolName}</strong><img src=\'{imgurl}\'
align=\'right\'></div></tpl>',
    itemSelector: 'div.tooltypelist',
    singleSelect: true,
    grouped: false,
    indexBar: false,
    scroll: true,
    });


    ExtMap.UI.toolList.on('itemtap', function (list, index, item, e) {
        listnode = list.store.getAt(index).get('toolx');
        console.log(list.store.getAt(index).get('toolx'));


        // Hvis "Near Point":
        if  (listnode == 1) {
            Ext.getCmp('locateid').hide();
            Ext.getCmp('toolid').hide();
            Ext.getCmp('submitPolyId').hide();
            Ext.getCmp('signaturid').hide();
            Ext.getCmp('addid').hide();
            Ext.getCmp('submitid').hide();
            Ext.getCmp('submitProxId').show();
            Ext.getCmp('spacerid').show();
            Ext.getCmp('cancelid').show();
            Ext.getCmp('choosemapid').hide();
            google.maps.event.addListenerOnce(mapdemo.map, "mousedown", function(evt
) {
                        measurePoint(evt.latLng);
                        markerLat = evt.latLng.lat();
                        markerLng = evt.latLng.lng();
                        console.log(evt.latLng);

                    });
            }

        // Hvis "Near Line":
        if  (listnode == 2) {
            Ext.getCmp('locateid').hide();
            Ext.getCmp('toolid').hide();
            Ext.getCmp('signaturid').hide();
            Ext.getCmp('addid').hide();
            Ext.getCmp('submitid').hide();
            Ext.getCmp('submitPolyId').show();
            Ext.getCmp('spacerid').show();
            Ext.getCmp('cancelid').show();
```

```
            Ext.getCmp('infoBarId').show();
            Ext.getCmp('submitProxId').hide();
            google.maps.event.addListener(mapdemo.map, "mousedown", function(evt) {
                        measureAdd(evt.latLng);
                        markerLat = evt.latLng.lat();
                        markerLng = evt.latLng.lng();

                    });
            }

        // Hvis "Within Polygon":
        if  (listnode == 3) {
            Ext.getCmp('locateid').hide();
            Ext.getCmp('toolid').hide();
            Ext.getCmp('submitPolyId').hide();
            Ext.getCmp('signaturid').hide();
            Ext.getCmp('addid').hide();
            Ext.getCmp('submitid').hide();
            Ext.getCmp('submitProxId').hide();
            Ext.getCmp('spacerid').show();
            Ext.getCmp('cancelid').show();
            Ext.getCmp('submitWithinId').show();
            google.maps.event.addListener(mapdemo.map, "mousedown", function(evt) {
                        measureAdd(evt.latLng);
                        markerLat = evt.latLng.lat();
                        markerLng = evt.latLng.lng();
                    });



            }

        // Hvis "Measure Length":
        if  (listnode == 4) {
            opacity = 0,
            Ext.getCmp('locateid').hide();
            Ext.getCmp('toolid').hide();
            //Ext.getCmp('choosemapid').hide();
            Ext.getCmp('signaturid').hide();
            Ext.getCmp('addid').hide();
            Ext.getCmp('submitid').hide();
            Ext.getCmp('submitPolyId').hide();
            Ext.getCmp('spacerid').show();
            Ext.getCmp('cancelid').show();
            Ext.getCmp('infoBar1Id').show();
            Ext.getCmp('submitProxId').hide();
            google.maps.event.addListener(mapdemo.map, "mousedown", function(evt) {
                        // When the map is clicked, pass the LatLng obect to the
measureAdd function
                        measureAdd(evt.latLng);
                    });
            }
```

```
        // Hvis "measure area":
        if  (listnode == 5) {
            opacity = 0.25,
            Ext.getCmp('locateid').hide();
            Ext.getCmp('toolid').hide();
            //Ext.getCmp('choosemapid').hide();
            Ext.getCmp('signaturid').hide();
            Ext.getCmp('addid').hide();
            Ext.getCmp('submitid').hide();
            Ext.getCmp('submitPolyId').hide();
            Ext.getCmp('spacerid').show();
            Ext.getCmp('cancelid').show();
            Ext.getCmp('infoBarId').show();
            Ext.getCmp('submitProxId').hide();
            google.maps.event.addListener(mapdemo.map, "mousedown", function(evt) {
                        // When the map is clicked, pass the LatLng obect to the
measureAdd function
                        measureAdd(evt.latLng);
                    });
            }

        // Hvis "Get coordinates":
        if  (listnode == 6) {
            google.maps.event.addListenerOnce(mapdemo.map, "mousedown", function(evt
) {

                        markerLat = evt.latLng.lat();
                        markerLng = evt.latLng.lng();
                        Ext.Msg.alert('Coordinates', 'lat: ' + markerLat.toFixed(6)
+'<br> lng: '+markerLng.toFixed(6), Ext.emptyFn);
                    });



            }

                ExtMap.UI.toolListOverlay.hide()
    });









    ExtMap.UI.toolListOverlay = new Ext.Panel({
    floating: true,
    modal: true,
    centered: false,
    bodyPadding: 3,
    bodyMargin: 3,
    width: Ext.is.Phone ? 222 : 222,
```

```
    height: Ext.is.Phone ? 220 : 220,
    scroll: false,
    items: [ExtMap.UI.toolList]
    });

    ExtMap.UI.showToolListOverlay = function (btn, event) {
        ExtMap.UI.toolListOverlay.setCentered(false);
        ExtMap.UI.toolListOverlay.showBy(btn)
    };

//-----------------------------------------------------------------------------
--------------
//                                  ToolList Slut
//-----------------------------------------------------------------------------
--------------




//-----------------------------------------------------------------------------
------------
//  Laver en "ListStore" indeholdende elementerne fra Locate-listen defineret i
toppen af koden
//-----------------------------------------------------------------------------
------------

Ext.regModel('locateList', {
        fields: ['locateName', 'imgurl']
    });

    ExtMap.UI.locateListStore = new Ext.data.Store({
    model: 'locateList',
    getGroupString: function (record) {
        return record.get('locateName')[0]
    },
    data: ExtMap.Locate.locateListData
    });

    ExtMap.UI.locateList = new Ext.List({
    width: Ext.is.Phone ? undefined : 200,
    height: 90,
    store: ExtMap.UI.locateListStore,
    itemTpl: '<tpl for="."><div
class="locatetypelist"><strong>{locateName}</strong><img src=\'{imgurl}\'
align=\'right\'></div></tpl>',
    itemSelector: 'div.locatetypelist',
    singleSelect: true,
    grouped: false,
    indexBar: false,
    scroll: false,
    });
```

```
    ExtMap.UI.locateList.on('itemtap', function (list, index, item, e) {
        listnode = list.store.getAt(index).get('locatex');
        console.log(list.store.getAt(index).get('locatex'));


        // Hvis "Current location":
        if  (listnode == 1) {
                        var geo = new Ext.util.GeoLocation({
                        autoUpdate: false,
                        allowHighAccuracy: false,
                        //accuracy: 1,
                        //altitude: 1,
                        timeout: 600000,
                        listeners: {
                            locationupdate: function (geo) {
                                mapdemo.map.setCenter(new google.maps.LatLng(geo.
latitude, geo.longitude));
                                mapdemo.map.setZoom(15);
                                console.log("lat.",geo.latitude, "lng.",geo.
longitude)
                                console.log(geo.accuracy, "meters")
                                console.log(geo.timestamp.d)

                                    //--add marker--
                                Marker = new google.maps.Marker({
                                    icon: 'img/marker_blue.png',
                                    animation: google.maps.Animation.BOUNCE,
                                    map: mapdemo.map,
                                    position: new google.maps.LatLng(geo.latitude,
geo.longitude),
                                });
                                markersArray.push(Marker); //Her pushes markers ind
i markersArray

                                var geolat = geo.latitude
                                var geolng = geo.longitude
                                var geoaccu = geo.accuracy

                                Ext.Msg.alert('Coordinates', 'lat.' + geolat.toFixed
(6)+'<br> lng. '+geolng.toFixed(6) + '<br> accuracy. ' + geoaccu + ' m', function ()
 {window.setTimeout(clearAll,5000)});

                            },
                        }
                    });

                    geo.updateLocation()
                };



        // Hvis "Find Address":
        if  (listnode == 2) {
```

```
                ExtMap.Utils.findAddress = function (map) {
                    Ext.Msg.prompt("Find Address", "Please enter an address",
function (value, txtvalue) {
                    console.log(value);
                    console.log(txtvalue);
                        ExtMap.Map.geocoder.geocode({
                            'address': txtvalue
                        }, function (results, status) {
                            if (status == google.maps.GeocoderStatus.OK) {
                                mapdemo.map.setCenter(results[0].geometry.location);
                                ExtMap.Map.addressMarker = new google.maps.Marker({
                                    icon: 'img/marker_blue.png',
                                    animation: google.maps.Animation.BOUNCE,
                                    map: mapdemo.map,
                                    position: results[0].geometry.location
                                });
                                mapdemo.map.setZoom(15);
                                console.log(results[0].geometry.location);
                                setTimeout(function () {
                                     ExtMap.Map.addressMarker.setMap(null)
                                 }, 5000)
                            } else {
                                Ext.Msg.alert('Alert', 'Geocoder failed due to: ' +
status, null)
                            }
                        })
                    }
                    ,
                    this,
                    false,
                    null,
                    {maxlength: 100, autocapitalize : false, placeholder :
"Address, town, country..."}
                    )
                };



                ExtMap.Utils.findAddress(mapdemo.map)



            };

        ExtMap.UI.locateListOverlay.hide()
    });



    ExtMap.UI.locateListOverlay = new Ext.Panel({
    floating: true,
    modal: true,
    centered: false,
```

```
    bodyPadding: 3,
    bodyMargin: 3,
    width: Ext.is.Phone ? 222 : 222,
    height: Ext.is.Phone ? 120 : 120,
    //dockedItems: ExtMap.UI.overlayLocateTb,
    //scroll: 'vertical',
    scroll: false,
    items: [ExtMap.UI.locateList]
    });

    ExtMap.UI.showLocateListOverlay = function (btn, event) {
        ExtMap.UI.locateListOverlay.setCentered(false);
        //ExtMap.UI.overlayLocateTb.setTitle('Locate');
        ExtMap.UI.locateListOverlay.showBy(btn)
    };
//-----------------------------------------------------------------------------
------------
//                                  LocateList Slut
//-----------------------------------------------------------------------------
------------


//  =============   GEM kortcenter, zoom og id i LOKOAL HUKOMMELSE
=============
    Ext.regModel('extmapLastPosition', {
    fields: [{
        name: 'id',
        type: 'int'
    }, {
        name: 'lat'
    }, {
        name: 'lng'
    }, {
        name: 'zoom',
        type: 'int'
    }, {
        name: 'maptype',
        type: 'string'
    }]
});

    ExtMap.Storage.positionStore = new Ext.data.Store({
    model: 'extmapLastPosition',
    proxy: {
        type: 'localstorage',
        id: 'positionStorage',
        proxy: {
            idProperty: 'id'
        }
    }
    });

    ExtMap.Storage.saveLastPosition = function (map) {
```

```
    ExtMap.Storage.positionStore.getProxy().clear();
    var newPosition = Ext.ModelMgr.create({
        id: 1,
        lat: map.getCenter().lat(),
        lng: map.getCenter().lng(),
        zoom: map.getZoom(),
        maptype: map.getMapTypeId()
    }, 'extmapLastPosition');
    ExtMap.Storage.positionStore.add(newPosition);
    ExtMap.Storage.positionStore.sync()
};

    ExtMap.Storage.loadPreviousPosition = function () {
    var positionArray = [];
    ExtMap.Storage.positionStore.load({
        scope: this,
        callback: function (operation) {
            if (operation.length == 0) {
                positionArray[0] = 55.676033;
                positionArray[1] = 12.568923;
                positionArray[2] = 12;
                positionArray[3] = "ROADMAP"
            } else {
                positionArray[0] = operation[0].data.lat;
                positionArray[1] = operation[0].data.lng;
                positionArray[2] = operation[0].data.zoom;
                positionArray[3] = operation[0].data.maptype
            }
        }
    });
    return positionArray
};
```

Bilag 4: dbinfo.php

```php
<?php
$username="postgres";
$password="******";
$database="geoDB";
$ip="localhost";
?>
```

Bilag 5: server.php

```php
<?php
    require("dbinfo.php");
     $conn_string = "host=$ip port=5432 dbname=$database user=$username password=
$password";
     $dbconn = pg_connect($conn_string);


    $date = $_POST['date'];
    $username = $_POST['name'];
    $type = $_POST['type'];
    $comment = $_POST['comment'];
    $lat = $_POST['lat'];
    $lng = $_POST['lng'];


    $post = pg_query($dbconn, "INSERT INTO temp_fortidsminder (timeof_cre ,
username, feat_type, comment, the_geom) VALUES ('$date', '$username', '$type', '
$comment', ST_Transform(ST_GeomFromText('POINT($lng $lat)', 4326), 25832))");
        if (!$post) {
            echo "An error occured.";
        }else {
            echo 'Success!!!!!!!!';
        }
?>
```

```php
<?php
   //---------------------------------------------------------------
   //       Opretter forbindelse til databasen
   //---------------------------------------------------------------
    require("dbinfo.php");
     $conn_string = "host=$ip port=5432 dbname=$database user=$username password=
$password";
     $dbconn = pg_connect($conn_string);

   //---------------------------------------------------------------
   //       Henter data fra submitformen og gemmer i variable
   //---------------------------------------------------------------
    $date = $_POST['date'];
    $username = $_POST['name'];
    $type = $_POST['type'];
    $comment = $_POST['comment'];
    $array = $_POST['array'];

   //---------------------------------------------------------------
   //   Laver teksstreng polygonens koordinater (form:"lat,lng,lat,lng,lat,lng...")
og tilføjer første punkt som sidste punkt.
   //---------------------------------------------------------------

    // fjerner parenteser og mellemrum fra array'et (tekststreng - filen ser
efterfølgende således ud: "lat,lng,lat,lng....")
    $array = str_replace("(", "", $array);
    $array = str_replace(")", "", $array);
    $array = str_replace(" ", "", $array);
    // laver variabel med første punkt (første og sidste punkt skal være det samme
ved indsættelse i databasen - NB! sådan er det ikke i Googles MVC-array)
    $explode = explode(",",$array);
    $string = "$explode[0],$explode[1]";

    // tilføjer første punkt sidst i det bearbejdede Google-array (tekststreng)
    //$array = "$array,$string";

    // omdanner ovenstående teksstreng til et array:
    $array = explode(",",$array);

   //---------------------------------------------------------------
   //   Laver løkke som bytter om på lat og lng (krav for at koordinaterne kommer
rigtig ind i databasen)
   //---------------------------------------------------------------
    $i = 0;     // variable som skal bruges til at få løkken til at køre
    $s= count($array);  // variable som angiver løkkens stoppunkt (antallet af
elementer i $array)
    $line= "";  // tom variabel som fyldes med koordinater når løkken kører.

    for ($i=0;$i<=$s-1; $i++)  {
        $lat = $array[$i];
        $i++;
        $lng = $array[$i];
```

```
        $line= "$line,$lng $lat";
    };

   //---------------------------------------------------------------
   //   Klargøre $line til indsættelse i database
   //---------------------------------------------------------------
    $line= substr($line,1);     // fjerner komma foran første koordinat

    $line= "($line)";       // tilføjer parentes om tekst    input til database
"POLYGON((lng lat, lng lat, lng lat"))

   //---------------------------------------------------------------
   //   Tilføjer polygon og øvrige oplysninger til databasen
   //---------------------------------------------------------------
    $post = pg_query($dbconn, "INSERT INTO temp_beskyt_linje (date, username,
objekttkst, comment, the_geom) VALUES ('$date', '$username', '$type', '$comment',
ST_Transform(ST_GeomFromText('MULTILINESTRING($line)', 4326), 25832))");
        if (!$post) {
            echo "An error occured.";
        }else {
            echo 'Success!!!!!!!!';
        };
 ?>
```

```php
<?php

   //---------------------------------------------------------------
   //       Opretter forbindelse til databasen
   //---------------------------------------------------------------
    require("dbinfo.php");
     $conn_string = "host=$ip port=5432 dbname=$database user=$username password=
$password";
     $dbconn = pg_connect($conn_string);

   //---------------------------------------------------------------
   //       Henter data fra submitformen og gemmer i variable
   //---------------------------------------------------------------
    $date = $_POST['date'];
    $username = $_POST['name'];
    $type = $_POST['type'];
    $comment = $_POST['comment'];
    $array = $_POST['array'];

   //---------------------------------------------------------------
   //   Laver teksstreng polygonens koordinater (form:"lat,lng,lat,lng,lat,lng...")
og tilføjer første punkt som sidste punkt.
   //---------------------------------------------------------------

    // fjerner parenteser og mellemrum fra array'et (tekststreng - filen ser
efterfølgende således ud: "lat,lng,lat,lng....")
    $array = str_replace("(", "", $array);
    $array = str_replace(")", "", $array);
    $array = str_replace(" ", "", $array);
    // laver variabel med første punkt (første og sidste punkt skal være det samme
ved indsættelse i databasen - NB! sådan er det ikke i Googles MVC-array)
    $explode = explode(",",$array);
    $string = "$explode[0],$explode[1]";

    // tilføjer første punkt sidst i det bearbejdede Google-array (tekststreng)
    $array = "$array,$string";

    // omdanner ovenstående teksstreng til et array:
    $array = explode(",",$array);

   //---------------------------------------------------------------
   //   Laver løkke som bytter om på lat og lng (krav for at koordinaterne kommer
rigtig ind i databasen)
   //---------------------------------------------------------------
    $i = 0;     // variable som skal bruges til at få løkken til at køre
    $s= count($array);  // variable som angiver løkkens stoppunkt (antallet af
elementer i $array)
    $polygon= "";   // tom variabel som fyldes med koordinater når løkken kører.

    for ($i=0;$i<=$s-1; $i++)  {
        $lat = $array[$i];
        $i++;
        $lng = $array[$i];
```

```
        $polygon= "$polygon,$lng $lat";
    };

   //-----------------------------------------------------------------
   //   Klargøre $polygon til indsættelse i database
   //-----------------------------------------------------------------
    $polygon= substr($polygon,1);   // fjerner komma foran første koordinat

    $polygon= "($polygon)";     // tilføjer parentes om tekst    input til database
"POLYGON((lng lat, lng lat, lng lat"))

   //-----------------------------------------------------------------
   //   Tilføjer polygon og øvrige oplysninger til databasen
   //-----------------------------------------------------------------
    $post = pg_query($dbconn, "INSERT INTO temp_paragraf3 (date, username,
objekttkst, comment, the_geom) VALUES ('$date', '$username', '$type', '$comment',
ST_Transform(ST_GeomFromText('MULTIPOLYGON(($polygon))', 4326), 25832))");
        if (!$post) {
            echo "An error occured.";
        }else {
            echo 'Success!!!!!!!!';
        };
 ?>
```

Bilag 8: serverRadius.php

```
<?php
    require("dbinfo.php");
     $conn_string = "host=$ip port=5432 dbname=$database user=$username password=
$password";
     $dbconn = pg_connect($conn_string);

    $search = array();


    $lat = $_GET['lat'];
    $lng = $_GET['lng'];
    $radius = $_GET['radius'];
    $type = $_GET['type'];


    //If statement der gør det muligt at vælge alle typer af punkter.
    if ($type == "skoler"){
    $post = pg_query($dbconn,
    "SELECT * FROM skoler WHERE ST_DWithin(ST_Transform(the_geom,
25832),ST_Transform(ST_GeomFromText('POINT($lng $lat)', 4326), 25832), $radius)");
    }
    else{
    $post = pg_query($dbconn,
    "SELECT * FROM fortidsminder WHERE ST_DWithin(ST_Transform(the_geom,
25832),ST_Transform(ST_GeomFromText('POINT($lng $lat)', 4326), 25832), $radius)");
    };


    while ($row =pg_fetch_object($post))  {
        $search[] = $row;
    };

echo json_encode(array('search' =>$search));
?>
```

```php
<?php
    require("dbinfo.php");
     $conn_string = "host=$ip port=5432 dbname=$database user=$username password=
$password";
     $dbconn = pg_connect($conn_string);

    $search = array();


   //---------------------------------------------------------------------
   //       Henter data fra submitformen og gemmer i variable
   //---------------------------------------------------------------------
    $type = $_GET['type'];
    $array = $_GET['array'];

   //---------------------------------------------------------------------
   //   Laver teksstreng polygonens koordinater (form:"lat,lng,lat,lng,lat,lng...")
og tilføjer første punkt som sidste punkt.
   //---------------------------------------------------------------------

    // fjerner parenteser og mellemrum fra array'et (tekststreng - filen ser
efterfølgende således ud: "lat,lng,lat,lng....")
    $array = str_replace("(", "", $array);
    $array = str_replace(")", "", $array);
    $array = str_replace(" ", "", $array);
    // laver variabel med første punkt (første og sidste punkt skal være det samme
ved indsættelse i databasen - NB! sådan er det ikke i Googles MVC-array)
    $explode = explode(",",$array);
    $string = "$explode[0],$explode[1]";

    // tilføjer første punkt sidst i det bearbejdede Google-array (tekststreng)
    $array = "$array,$string";

    // omdanner ovenstående teksstreng til et array:
    $array = explode(",",$array);

   //---------------------------------------------------------------------
   //   Laver løkke som bytter om på lat og lng (krav for at koordinaterne kommer
rigtig ind i databasen)
   //---------------------------------------------------------------------
    $i = 0;     // variable som skal bruges til at få løkken til at køre
    $s= count($array);  // variable som angiver løkkens stoppunkt (antallet af
elementer i $array)
    $polygon= "";   // tom variabel som fyldes med koordinater når løkken kører.

    for ($i=0;$i<=$s-1; $i++)  {
        $lat = $array[$i];
        $i++;
        $lng = $array[$i];

        $polygon= "$polygon,$lng $lat";
    };
```

```
   //---------------------------------------------------------------------
   //   Klargøre $polygon til indsættelse i database
   //---------------------------------------------------------------------

   $polygon= substr($polygon,1);    // fjerner komma foran første koordinat

    $polygon= "($polygon)";     // tilføjer parentes om tekst    input til database
"POLYGON((lng lat, lng lat, lng lat"))



    $post = pg_query($dbconn,
    "SELECT * FROM $type WHERE ST_Intersects(ST_Transform(the_geom,
25832),ST_Transform(ST_GeomFromText('MULTIPOLYGON(($polygon))', 4326), 25832))");


    while ($row =pg_fetch_object($post))  {
        $search[] = $row;

    };

    echo json_encode(array('search' =>$search));

?>
```

Bilag 10: Test, Positionsbestemmelse

punkt	lat	lng	type	afvigelse
1	**55,73233**	**12,39022**	**Garmin**	
1	55,732273	12,390152	iPhone	13,55
1	55,732286	12,390166	Locate	10,56
1	55,732337	12,390224	Get XY	1,45
2	**55,73167**	**12,39166**	**Garmin**	
2	55,731647	12,391636	iPhone	5,27
2	55,731522	12,391837	Locate	35,22
2	55,731649	12,391627	Get XY	5,54
3	**55,73332**	**12,3925**	**Garmin**	
3	55,733307	12,392479	iPhone	3,47
3	55,733314	12,392533	Locate	3,86
3	55,733281	12,392512	Get XY	7,81
4	**55,73462**	**12,39234**	**Garmin**	
4	55,73455	12,392321	iPhone	13,97
4	55,734553	12,392205	Locate	20,01
4	55,734554	12,392327	Get XY	13,1
5	**55,73507**	**12,39056**	**Garmin**	
5	55,735100	12,390559	iPhone	5,92
5	55,735977	12,390555	Locate	2,62
5	55,735020	12,390514	Get XY	11,11
6	**55,73605**	**12,38861**	**Garmin**	
6	55,736042	12,388641	iPhone	3,79
6	55,735977	12,388582	Locate	14,73
6	55,736045	12,388556	Get XY	6,09
7	**55,73320**	**12,38627**	**Garmin**	
7	55,733185	12,386258	iPhone	3,47
7	55,733173	12,386227	Locate	7,16
7	55,733197	12,386256	Get XY	1,67
8	**55,72460**	**12,38522**	**Garmin**	
8	55,724537	12,385403	iPhone	18
8	55,724557	12,385417	Locate	14,26
8	55,724558	12,385488	Get XY	9,01
9	**55,72234**	**12,38478**	**Garmin**	
9	55,722275	12,384954	iPhone	22,23
9	55,722274	12,384955	Locate	23,43
9	55,722317	12,384768	Get XY	4,72

Gennemsnitlig afvigelse:

iPhone	10,0
Locate	14,7
Get XY	6,7