Hovedprosjekt 2016

Høgskolen i Oslo og Akershus

Fakultet for kunst, teknologi og design

Store Finder Norway

Hybrid-webapplikasjon for søk av lokasjoner

Gruppe 31

Trung van Trinh s188085

Igor Cirkovic s178273

MediaHagen

HØGSKOLEN I OSLO
OG AKERSHUS

HØGSKOLEN I OSLO OG AKERSHUS

Studieprogram: Informasjonsteknologi
Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo
Besøksadresse: Holbergs plass, Oslo

Telefon: 22 45 32 00
Telefaks: 22 45 32 05

PROSJEKT NR.
2016 - 31

TILGJENGELIGHET
Åpen

BACHELORPROSJEKT

HOVEDPROSJEKTETS TITTEL	DATO
Store Finder Norway Hybrid-webapplikasjon for søk av lokasjoner	24.05.2016
	ANTALL SIDER / BILAG 97
PROSJEKTDELTAKERE	INTERN VEILEDER
Trinh van Trung – s188085 Igor Cirkovic – s178273	Geir Skjevling

OPPDRAGSGIVER	KONTAKTPERSON
Media Hagen Billingstadåsen 22B, 1396 Billingstad www.mediahagen.no	Harald S. Adamsen Tlf: 90 55 17 12 e-post: harald@mediahagen.no

SAMMENDRAG

Applikasjonen skal gjøre det enkelt for brukere å finne fram til ønskede butikker rundt om i Norge.
Den kan kjøres på web, mobil eller nettbrett.

Utviklet for Media Hagen.

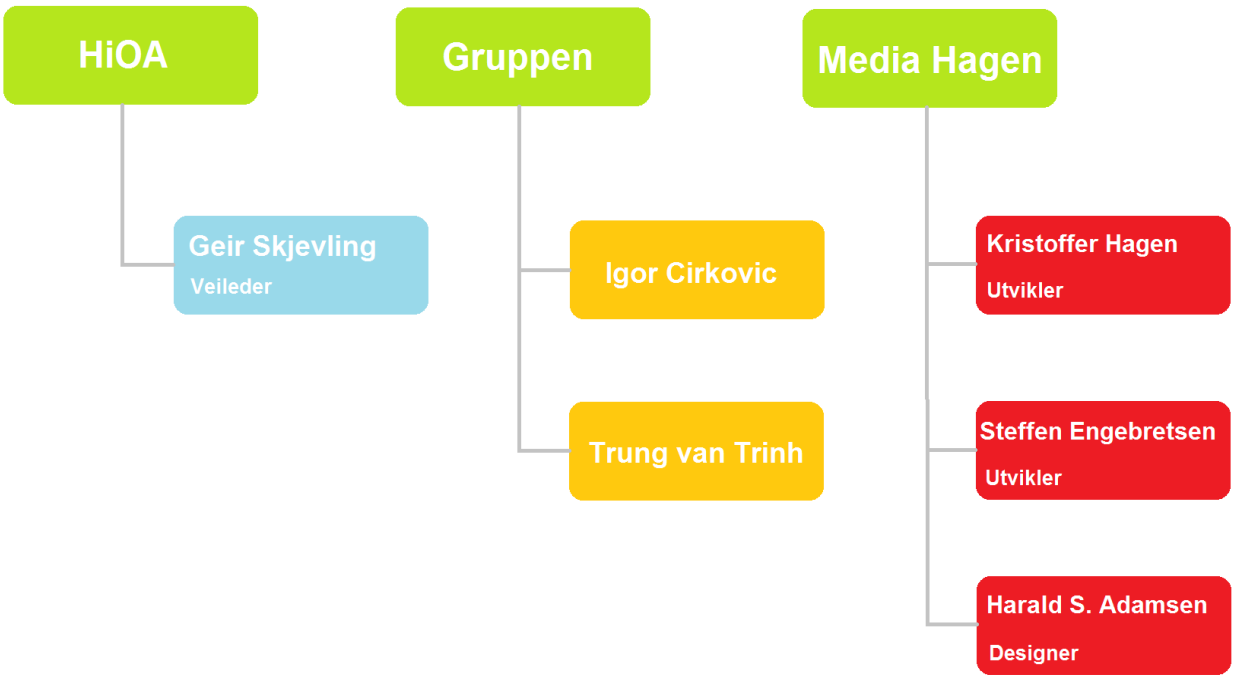
3 STIKKORD
Brukervennlighet
Tilgjengelighet
Apache Cordova

INNHOLDSFORTEGNELSE

Del 1 - Innledning

GRUPPEN

Gruppen består av to tredjeårsstudenter ved Høgskolen i Oslo og Akershus. Trinh van Trung studerer dataingeniørfag, og Igor Cirkovic studerer informasjonsteknologi. Gruppemedlemmene har ikke jobbet sammen tidligere før hovedprosjektet startet, og gruppen ble satt sammen over studentmailen i høst.



Figur 1.1 - Organisasjonskart

OPPDRAGSGIVER

Media Hagen er en liten IT-bedrift som spesialiserer seg på utvikling av applikasjoner og nettsider for både store og små bedrifter. Firmaet startet opp i høsten 2015 og består av Kristoffer Hagen, Harald S. Adamsen, og Steffen Engebretsen - alle tidligere HiOA-studenter.

De grunnleggende verdiene til selskapet er brukervennlighet, universell utforming, søkbarhet og et godt visuelt uttrykk. Disse er de sentrale temaene som legges vekt på når de utvikler IT-løsninger. Løsningene skal altså gjøres enkle å bruke, være tilgjengelig for alle uansett kunnskapsnivå og funksjonsnedsettelser, ha god kompatibilitet med mobile enheter, og ha et fint design som gjenspeiler det visuelle uttrykket til kunden/bedriften. Dette er noe vi selv har fokusert på siden vi fikk oppgaven og forsøkt å gjengi i vår egen applikasjon.

Media Hagen vil bli referert til som "arbeidsgiver" videre i dokumentasjonen av dette prosjektet.

OPPGAVEN

"Utvikle en nettside som gir brukeren informasjon om hvilke butikker som er i nærheten. Dette prosjektet krever kunnskaper i HTML/CSS/PHP/JSON/JavaScript/MySQL. Vi har en fungerende demo som finner ut hvor brukeren er, regner ut avstanden til de ulike butikkene via Google API og sorterer etter avstand i minutter/kilometer. Denne ønskes videreutviklet. Hovedfokuset her er at siden skal være så enkel som mulig å bruke, med spesielt fokus på mobil og nettbrett."

Etter en del intern diskusjon i gruppen kom vi fram til at det var best å utvikle en egen applikasjon fra bunnen, noe arbeidsgiver gikk med på. Dette gjorde at vi fikk full kontroll over rammeverket og kunne legge opp til videreutvikling. Vi fikk frie tøyler til å utvikle appen slik vi trodde var best.

MÅL

Applikasjonen er ment for å gi brukere mulighet til å få en enkel oversikt over butikker i nærheten, en måte å søke opp spesifikke butikker - og hvordan komme seg fram til dem ved hjelp av bil, buss, gange osv.

Vi ønsker å takke vår veileder Geir Skjevling og arbeidsgiver Media Hagen; Kristoffer Hagen, Harald S. Adamsen og Steffen Engebretsen.

Vi vil også takke lærerne vi har hatt gjennom årene på høyskolen: Eva Hadler Vihovde (programmering/programutvikling), Torunn Gjester (databaser/android), Thor Krattebøl (webprogrammering/webapp), og Siri Fagernes (webprosjekt). Dere har gitt oss kunnskapen og innblikket vi trengte for å kunne gjennomføre denne bacheloroppgaven.

Del 2 - Kravspesifikasjon

BAKGRUNN

Dette er kravspesifikasjonen som er utarbeidet for vårt hovedprosjekt våren 2016 ved Høgskolen i Oslo og Akershus. Hensikten med denne kravspesifikasjonen er å sikre en god forståelse over hvilke brukerfunksjoner som skal inkluderes i løsningen, hvordan brukergrensesnittet skal se ut og hvilke andre behov arbeidsgiver skulle ha som må dekkes for å oppnå et tilfredsstillende sluttresultat. Dette dokumentet ble opprettet i startfasen av prosjektet, men ble gjort en del endringer på underveis ettersom prioriteringene våre endres, og ble ferdigstilt mot avslutningsfasen av prosjektet. Her dekkes hvilke krav som er satt for systemet og en beskrivelse av hvordan produktet burde se ut når utviklingen er avsluttet.

SYSTEMBESKRIVELSE OG KRAV

Kravspesifikasjonen definerer betingelser og skal være til hjelp for gruppe, veileder og arbeidsgiver for å fjerne enhver tvil over hva som skal lages. Dette har vært til god nytte for gruppen i situasjoner hvor vi har vært usikre på hva som skal inkluderes av funksjonalitet og hvordan vi skal arbeide i veien fremover.

ARBEIDSGIVERS KRAV

Kravspesifikasjonen vi fikk av arbeidsgiver var noe kort og dekket kun de sentrale funksjonene applikasjonen trengte. Siden vi fikk frie tøyler utarbeidet vi vår egen kravspesifikasjon som er

mer detaljert og spesifikk på hvilke mål vi setter for oss selv for å kunne opprettholde en oversikt over hva den ferdigstilte løsningen burde inkludere av funksjonalitet, brukervennlighet og design. Punktene vi fikk fra arbeidsgiver var:

- Gi bruker informasjon over hvilke butikker som er i nærheten
- Gi bruker mulighet til å søke opp butikker
- Gi bruker informasjon over hvor langt unna lokasjonen befinner seg
- Gi bruker informasjon over bedrifters åpningstider, navn og adresse
- Gjøre appen så enkel som mulig å bruke

Vi har som nevnt fått en stor grad av frihet til å utvikle applikasjonen slik vi vil, men disse punktene anses som veiledende for prosjektet. Vi har da naturligvis beholdt dem som kjernen i spesifikasjonen og lagt til nye i vår egen opparbeidede versjon for å tilpasse visjonen vi som gruppe hadde for prosjektet.

KRAVSPESIFIKASJONEN

PRIORITERT FUNKSJONALITET

Systemet skal kunne:

- Ha en "oppdater"-knapp som viser hvor brukeren befinner seg
- Ha en oversikt over butikker i nærheten av bruker
- Ha en søkemotor for å søke opp spesifikke lokasjoner
- Hente fram viktig informasjon om bedrifter, slik som åpningstider og adresse
- Beregne avstand til lokasjoner og vise hvordan man kommer dit ved hjelp av bil, gange og offentlig transport
- Ha en automatisk liste som ramser opp butikker i nærheten
- Være åpent: ingen innlogging fra brukerens side og ingen administrative funksjoner

ØNSKET FUNKSJONALITET

Systemet burde kunne:

- Vise anmeldelser for ulike bedrifter
- Ha en knapp for deling til sosial media

IKKE-FUNKSJONELLE KRAV

Brukervennlighet:

- Appen skal følge retningslinjer til universell utforming
- Appen skal ikke inneholde fagord eller vanskelig språk
- Språket skal være på norsk bokmål

Estetiske krav:

- Appen skal være så enkel som mulig å bruke og intuitiv, dvs. at det skal være tydelig hvordan man bruker den, og man skal ikke bruke lang tid på å finne det man er ute etter
- Appen skal starte opp med en fin bakgrunn
- Appen skal ha et gjentakende, konsistent fargetema

TEKNISKE KRAV

- Applikasjonen skal utvikles med Apache Cordova og bruke Google Maps API
- Applikasjonen skal kodes i hovedsakelig HTML5, CSS og JavaScript.
- Applikasjonen skal kunne kjøres på PC ved bruk av de mest kjente nettleserne (Internet Explorer, Google Chrome, Mozilla Firefox, Opera).
- Applikasjonen skal kunne kjøres på Android-enheter (operativsystem 4.2.2 og nyere).
- Applikasjonen skal kunne kjøres på mobile Windows-enheter.
- Applikasjonen skal kunne kjøres på iOS-enheter.
- Det skal være enkelt å navigere i applikasjonen og finne det man er ute etter.

- Applikasjonen skal laste raskt og være responsiv.

KRAV TIL KODE

Koden skal være ryddig og lett å forstå. Den skal ha en strukturert oversikt slik at Media Hagen uten vanskeligheter kan sette seg inn i den ved vedlikehold og eventuell videreutvikling. Navn på funksjoner og lignende bør være selvforklarende slik at det ikke er nødvendig med mye kommentarer. Kommentarer til kode skal være på norsk bokmål.

OPPSUMMERING

Kravspesifikasjonen har i stor grad vært til hjelp både i oppstarten av utviklingen og i gjennomføringsprosessen. I begynnelsen fikk vi definert prosjektets rammer og begrensninger, samtidig som vi fikk et overblikk over hva som var mest essensielt å fullføre først og hva vi kunne legge til side til senere. Spesifikasjonen ble fulgt nøye og bidro til strukturert arbeid, samt et passende produkt for arbeidsgiver.

Del 3 - Prosessrapport

I denne rapporten beskrives prosessen bak utviklingen av Store Finder Norway for Media Hagen i forbindelse med hovedprosjektet våren 2016. I denne delen tar vi for oss blant annet hvordan vi fant oppgaven, tanker vi hadde i starten av prosjektet og hvordan vi planla å gjennomføre det, og ikke minst hvordan selve utviklingsprosessen har foregått for vår gruppe.

PROSJEKTSTART

Gruppemedlemmene hadde som nevnt ikke jobbet sammen tidligere, og gruppen ble dannet over mail i starten av november 2015. Vi hadde et møte kort tid etterpå og satte ut for å finne en oppgave. Med tanke på at vi kun er to personer tenkte vi at ikke kunne begi oss på et hvilket som helst prosjekt, og prøvde derfor å finne et som var aktuelt for oss. Etter å ha prøvd oss på en simulator-app for oljeingeniørstudenter (arbeidsgiver Tina Komulainen) i desember, viste det seg at oppgaven ble for stor og vi prøvde å finne noe nytt.

Vi kom over oppgaven til Media Hagen på HiOAs prosjektforslagside i januar og fattet fort interesse for den. Etter å ha kommunisert med dem via e-post fikk vi eventuelt muligheten til å velge mellom to oppgaver; lage en nettside, eller lage en app for søking av butikker. Etter en kort diskusjon i gruppen ble vi enige om at det sistnevnte alternative passet best for oss, og vi fikk satt opp et møte med arbeidsgiver 26. Januar. På det første møtet viste de oss demoen for applikasjonen som de ønsket videreutviklet, men foreslo også at vi kunne lage en app fra bunnen av dersom det passet oss bedre. Vi ville da få frihet til å designe denne slik vi ville, bare de sentrale funksjonene ble beholdt. Etter dette møtet bestemte vi oss for å ta denne oppgaven, og dermed ble Media Hagen vår arbeidsgiver.

MILEPÆLSPLAN

Vi fikk ikke opprettet en milepælsplan med det første, men forsto underveis at dette ville være viktig for å kunne arbeide med prosjektet på en strukturert og kontrollert måte, samtidig som det kunne hjelpe oss med å opprettholde tidsfristen på konkrete delmål siden vi allerede var relativt sent ute med å finne et prosjekt. Planen var til god nytte for gruppen, og er vist under:

Dato	Framdrift	Mål til neste møte
Onsdag 27.01.2016	Begynner på nytt prosjekt. Starter med planlegging av møtetider og leser alt vi kan til neste møte.	
Tirsdag 2.2.2016	Legger fram ideer om hvordan vi skal gå fram med oppgaven.	
Fredag 5.2.2016	Lager tegning og bestemmer hva slags utviklingsverktøy som er nødvendig for å arbeide med prosjektet.	Last ned all nødvendig programvare til egen bærbar datamaskin.
Tirsdag 8.2.2016	Sette opp verktøy for bruk til privat PC: NetBeans 8.1 med Cordova/PhoneGap.	
Mandag 15.2.2016	Har ikke fått satt opp verktøyene enda. Møtt på et par installasjonsproblemer og misforståelser med arbeidsgiver angående oppgaven.	Alt stoppes opp.
Tirsdag 16.02.2016	Prosjektet startes igjen.	Til neste uke må alt installeres. Begynne med koding.

Fredag 23.02.2016	Fått installert inn verktøy på egne maskiner: NetBeans 8.1 med Cordova, Android SDK Bundle, Windows Phone SDK, Ant build, Git.	Lese alt om Google Maps API JavaScript og prøve koden selv.
Tirsdag 1.03.2016	Implementere demo av Google Maps API JavaScript i fellesskap og snakke om hvordan koden fungerer.	
Fredag 4.03.2016	Begge jobber.	Hjemmearbeid; lage en egen Google Maps hver for seg og lage tegning til neste møte.
Tirsdag 8.03.2016	Problem med whitelist, skjermen vil ikke kjøre ordentlig til Android og gir kun en blank skjerm.	Fikse whitelist-problemet til neste møte.
Fredag 18.03.2016	Fått løst problemet ved å sette ajax = false. Tar igjen tapt tid ved å sitte hele dagen ut til 22.	Implementere sammen ting vi har kommet fram til.
Tirsdag 22.03.2016	Sitter sammen og justerer på hvordan ting skal ligge på map og hva slags funksjoner som skal være med. Sendte mail til arbeidsgiver - spurte om kravspesifikasjon og hva de synes.	Implementerer inn Google service til map. Møtes ikke før 29.03.
Tirsdag 29.03.2016	Et medlem av gruppa er syk. Jobbes da hjemmefra hver for seg.	Forsøke å fikse geolocations automatiske oppdatering til neste møte.
Fredag 01.04.2016	Geolocations er oppe med søkemotor. Tar tidlig helg.	Finne på hva som skal gjøres til neste gang, Transport/Directions Google Maps API.

Tirsdag 05.04.2016	Prosjektarbeidet utsettes til neste uke. Masse innleveringer i andre fag som må prioriteres.	Fortsatt samme mål.
Fredag 08.04.2016	Sitter sammen dagen ut. Ekstra møte på søndag for å bli ferdig.	Implementere og fikse på buttons i reise.js
Tirsdag 12.04.2016	Dropper møtet i dag, har ikke kommet fram til noen bedre løsning. Vi prøver å løse det hver for oss hjemmefra.	Fortsatt fikse radio-buttons i reise.js. Vil ikke forandres når man prøver å bytte.
Tirsdag 19.04.2016	Har enda ikke fått fikset knappeproblemet. Vi går over til noe nytt og forsøker igjen når vi kommer på noe smart.	Prøve å lage en offline lagring uten å bruke internett.
Fredag 22.04.2016	Vi dropper offline lagring. Lager nå en funksjon så folk kan finne fram uten tilgang til internett.	Implementere inn design og hvordan den skal se ut.
Tirsdag 26.04.2016	Jobb og innleveringer av obliger i andre fag.	
Fredag 29.04.2016	Nå har vi fått lagt inn alt som trengs på den siste delen.	Fikse CSS.
Fredag 06.05.2016	Begge har vært syke hele uken.	
Tirsdag 10.05.2016	Sjokk!! Levering om 14 dager, er enda ikke på god vei med dokumentasjonen.	Deler oppgaven i to. Setter opp struktur for hvordan dokumentasjon skal se ut og hva som må legges inn. Så fikse på design og gjøre det fint.
Onsdag 11.05.2016	Så mye å gjøre at vi helt glemte å oppdatere dagbok. Dette blir siste dato for oppdatering.	Herfra vil vi ta det som vi rekker.

Vi førte også en dagbok over gruppemøter underveis i semesteret, hvor vi i korthet beskriver når vi møttes, hva vi gjorde og hva vi planlegger til neste gang. Denne ble etter vår mening overflødig etter hvert ettersom vi hadde god kommunikasjon med hverandre jevnt og trutt.

RISIKOPLAN

Under er en risikoplan som er en oversikt over hva som kan gå galt i utviklingen, og følgene det kan få. Her beskrives hvordan vi skal forebygge disse hendelsene, og tiltak for problemer hvis de skulle oppstå.

Mulige risikoer	**Sannsynlighet**	**Konsekvens**
Sykdom	Stor	Middels
Uenighet i gruppen	Stor	Liten
Uenighet med arbeidsgiver	Middels	Stor
Manglende oppmøte	Middels	Middels
Tekniske problemer	Middels	Stor
Faglige problemer	Stor	Stor
Tap av data	Liten	Stor
Sette seg for vanskelige mål	Middels	Middels

Figur 3.1 – Risikoplan

BESKRIVELSE AV RISIKOENE

Sykdom er en ganske stor risiko for gruppen, og kan bli en stor konsekvens hvis begge gruppemedlemmene skulle bli syke samtidig. Sykdom kan inntreffe når som helst og vil føre til mindre tid og mer jobb for den friske parten.

Forebygging: Følge med på arbeidet til andre gruppemedlemmer.

Tiltak: God kommunikasjon så den som er syk kan jobbe hjemmefra. Hvis det ikke er aktuelt, omfordele oppgaver.

Uenighet i gruppen er noe som alltid kan oppstå hvis gruppemedlemmene har forskjellige ideer om hvordan arbeidet bør utføres eller ikke er fornøyde med innsatsen til hverandre.

Forebygging: Ha tett dialog.

Tiltak: Snakke tydeligere og mer åpent, behandle hverandre rettferdig og komme til enighet.

Uenighet med arbeidsgiver kan føre til forsinkelser i utviklingen eller i verste fall avvikling av hele prosjektet.

Forebygging: Ha tett og åpen dialog.

Tiltak: Snakke tydeligere og mer åpent, behandle hverandre rettferdig og komme til enighet.

Manglende oppmøte kan forsinke hele arbeidet, og en person må ta på seg mer ansvar/arbeid enn den andre. Dette fører til avvik fra planen vi har lagt for oss og kan også føre til konflikter.

Forebygging: Tydelige innkallelser.

Tiltak: Omfordele oppgaver, eventuelt utkastelse fra gruppen hvis det gjentar seg ofte.

Tekniske problemer som at en PC ikke fungerer som den skal eller at problemer med programvare kan oppstå kan vi ikke se bort fra.

Forebygging: Bruke riktig utstyr og programvare.

Tiltak: Prøve å rette opp feil, i verste fall bytte utstyr/programvare.

Faglige problemer vil kunne oppstå, i form at det er noe vi finner vanskelig med design, koding eller rent faglig.

Forebygging: Hjelpe hverandre, sette av god tid.

Tiltak: Finne en løsning i fellesskap. Bruke veileder eller andre kilder til hjelp.

Tap av data er alltid en risiko som kan medføre store tap av tid. Selv om det er liten sannsynlighet for at dette vil inntreffe vil det ha store konsekvenser for arbeidsprosessen og kan også gjøre at gruppen mister motivasjon.

Forebygging: Lagre ofte og ta backup, gjerne på flere steder (ekstern harddisk, Dropbox, Github).

Tiltak: Ta i bruk backup. I verste fall må arbeid gjøres om igjen.

Sette seg for vanskelige mål kan fort bli et problem når vi planlegger framover uten å ha et godt bilde over hvor mye tid og ressurser vi har til gode. Dette går hånd i hånd med tidsmangel, og det kan bli mye arbeid i perioder, spesielt i sluttfasen.

Forebygging: Planlegge godt og holde interne tidsfrister.

Tiltak: Nedprioritering av enkelte deler av prosjektet (eller droppe det helt i verste fall), omfordeling av arbeidsoppgaver.

ARBEIDSTEKNIKKER

Her vil vi beskrive hvordan arbeidet med prosjektet har foregått og hvordan kommunikasjon med veileder og arbeidsgiver har vært gjennom utviklingsperioden.

GRUPPEARBEID

Jobbing i team

Gruppen har gjennom semesteret fått erfare hvordan det er å jobbe i team og samarbeide mot et felles mål. Det har vært vanskelig å beregne tid og størrelse på prosjektet, spesielt når nye oppgaver konstant dukker opp og når man støter på feil som er frustrerende og videre begrenser tiden som vi har til gode. Prosjektarbeidet har vært en omfattende, men veldig lærerik prosess for gruppen.

Arbeidsforhold

Vi har for det meste holdt til i HiOAs lokaler i Pilestredet 35 (fakultet for teknologi, kunst og design). Her har vi benyttet oss av grupperom og stasjonære PCer i datalabene. Ellers har vi jobbet hjemme hver for oss.

Figur 3.2 – HiOA, Pilestredet 35

Jobb

Begge gruppemedlemmene har hatt en deltidsjobb ved siden av studiene, noe som har påvirket tidspunktene vi kunne møtes på skolen. Vi har vært fleksible og planlagt møtene rundt dette, og har vært forståelsesfulle ovenfor hverandre hvis noen ikke kunne komme i et møte.

KOMMUNIKASJON MED INTERN VEILEDER OG ARBEIDSGIVER

Kommunikasjon med intern veileder Geir Skjevling var stort sett mest utbredt i startfasen av prosjektet, når vi trengte hjelp til å velge oppgave og hvordan vi skulle gå fram med dokumenteringsprosessen. Siden han ikke hadde erfaring med applikasjonsutvikling kunne han ikke hjelpe oss med faglig innsikt. Veiledningen var med andre ord til prosjektstyring, ikke selve utviklingen av applikasjonen. Han var tilgjengelig når vi trengte han, til stor hjelp med å få oss til å starte med hovedprosjektet riktig, og vi tror han hadde en sterk innvirkning på å velge en oppgave som passet oss godt.

Gruppen møtte arbeidsgiver kun en gang i løpet av semesteret, som var når vi fikk oppgaven. På dette møtet fikk vi nok informasjon til å starte utviklingen av prosjektet, og siden har kommunikasjon mellom gruppen og arbeidsgiver foregått over e-post. Det oppsto noen misforståelser i starten av februar over hvilken oppgave vi valgte, og dette førte til en forsinkelse av utviklingen og ikke en like god startfase av arbeidet som vi håpet på. Vi fikk løst opp i dette og fortsatte arbeidet som planlagt. Misforståelser til side har kommunikasjonen vært profesjonell og de har vært flinke til å svare på spørsmål, men ikke veiledet oss i noen stor grad siden vi fikk såpass mye frihet til hvordan vi skulle utføre prosjektet.

UTVIKLINGSTEKNIKKER

Teknikken vi har tatt i bruk er smidig utvikling. Denne teknikken mente vi var best fordi vi følte at kravene vi hadde satt for oss selv ville komme til å endre seg underveis. Vi kunne ikke samle alle krav til en løsning før vi begynte utviklingen av applikasjonen, og de sentrale funksjonene måtte på plass før vi eventuelt kunne bygge videre og legge til nye funksjoner. Disse ville nok også endre seg, så vi måtte forutse at vi ikke hadde nok tid til å implementere alt vi ville, samtidig som vi måtte gjøre en del forandringer på det vi på forhånd hadde planlagt skulle bli en del av applikasjonen. Med andre ord prioriterte vi det viktigste først slik at vi fikk et fungerende program, og oppdaget veien videre ved diskusjon i gruppen og testing av ulike funksjoner.

Våre oppgaver som smidige utviklere ble raskt klart:

Kravanalyse og kundekontakt: Jobbe med å forsøke å forstå behovet til kunden (i vårt tilfelle, arbeidsgiver).

Prosjektplanlegging: Tenke smart i forhold til hva neste steg optimalt sett bør være - planlegge godt og loggføre alt for å få en god oversikt over prosessen for å bedre kunne avgjøre hva som må gjøres videre. Følge milepæls- og fremdriftsplaner.

Design og arkitektur: Tenke på hvordan løsningen bør designes for å være mest mulig fleksibel - legge et solid grunnlag for videreutvikling og gjøre løsningen enkel å sette seg inn i og bruke.

Prosjektstyring: Kommunisere med alle andre team-medlemmer om endringene som skjer - essensielt ved bruk av smidig utvikling, fordi ting fort kan endre seg og alle må hele tiden være på samme nivå for å kunne fremme arbeidet på en mest effektiv måte.

Implementasjon: Tilføre verdi gjennom å kode løsningen med så høy kvalitet som mulig - kvalitet over kvantitet. Gjennomføre en funksjon før vi beveger oss mot neste.

Kvalitetskontroll: Tenke kritisk og teste løsningen så effektivt som mulig.

(ref: kjempekjekt)

Metoden vi har brukt er en kombinasjon av Scrum og Kanban. Vi har jobbet i tette, løpende sprinter hvor vi har fokusert på å levere den viktigste funksjonaliteten først, samtidig som vi har unngått å forplikte oss til annet enn det vi har jobbet med i disse periodene. Fasene vi har gått gjennom er tradisjonelt for Scrum:

Planlegging: Skape en visjon for prosjektet. Definere behov og mål. Opprette en backlog; hva skal prioriteres først? Hvem skal være scrum master?

Gjennomføring: Definere spesifikke oppgaver og utføre dem. Opprette og bruke prosjektstyringsdokumenter flittig for å tilrettelegge en strukturert utviklingsfase av løsningen.

Avslutting: Evaluering og levering av løsningen.

Vi har også tatt i bruk grunnleggende prinsipper fra Kanban:

- Begrense antall oppgaver det arbeides med samtidig
- Teste eksisterende funksjoner og rette feil før nye funksjoner implementeres
- Fullstendig fullføre påbegynt arbeid før det avsluttes og en ny jobb startes
- Jevn flyt og kontinuerlig prioritering av planlagt arbeid

(ref: machina)

Gruppen er enig i at smidig utvikling definitivt har fungert godt for dette prosjektet, og denne metoden er noe vi begge har erfaring med fra tidligere prosjekter selv om vi ikke har samarbeidet tidligere. Det skal nevnes at vi droppet å utnevne en "scrum-master", siden vi er et team bestående av kun to personer, så det mente vi var unødvendig. Vi har heller ikke hatt daglige møter ("stand-up meetings") fordi vi har kommunisert jevnlig over internett og hele tiden vært klar over hvor den andre står. Vi har siden oppstart gjort gruppemøter til en fast rutine på tirsdager og fredager i løpet av semesteret.

VERKTØY

I denne delen utreder vi hva vi har brukt av utviklingsverktøy og kommunikasjonsmidler til å gjennomføre prosjektet.

KOMMUNIKASJONSMIDLER

E-post har blitt brukt som det primære kommunikasjonsverktøyet med intern veileder og spesielt arbeidsgiver, der vi har delt mindre filer.

Ved hjelp av studentmailen fikk vi også opprettet hovedprosjektgruppen.

Facebook er en nyttig sosial plattform som gjør det enkelt for mennesker å kommunisere med hverandre. Facebooks chat-funksjon har blitt brukt mye innad gruppen, både til samtaler om hva som må gjøres i prosjektet og til å sette opp møter.

DELING AV FILER

Dropbox er en amerikansk nettskytjeneste som synkroniserer filer mellom PC, nett og mobile enheter. Her har man muligheten til å opprette mapper og invitere andre personer til å bli med og dele filer, enten det er video, bilder eller dokumenter. I gruppen har vi brukt dette til å dele dokumenter og bilder som ble tatt i bruk i applikasjonen. Ulempen her er at flere personer ikke kan redigere dokumenter i sanntid.

Google Docs (Google Dokumenter) er en tjeneste som gjør at man kan opprette dokumenter på nett og enkelt samarbeide med andre. Her blir alle endringer lagret automatisk, og flere personer kan være inne og arbeide på samme dokument samtidig. Dette ble brukt flittig i gruppen i utarbeidingen av prosjektstyringsdokumentene.

Github er et web-basert hostingsystem for Git-prosjekter, og tilbyr versjonkontroll og administrering av kildekode. Her kan man dele kode og se hele historikken til hvordan den har endret seg, og gå tilbake til et hvilket som helst punkt det er gjort endringer.

Dette verktøyet gjør det lett for flere å samarbeide på den samme koden, og man unngår konflikter som kan oppstå ved å ha forskjellig kode på forskjellige maskiner.

UTVIKLING

Microsoft Word er et velkjent tekstbehandlingsprogram. Dette har vi brukt til å ferdigstille all dokumentasjon etter å ha fått inn rå tekst i Google Docs.

NetBeans (versjon 8.1) er en programvareutviklingsplattform. Den er primært ment for Java, men støtter også andre programmeringsspråk slik som C/C++, PHP og HTML5.

Apache Cordova (versjon 6.1.1) er et populært rammeverk for utvikling av mobile applikasjoner. Måten Cordova fungerer på er at det "wrapper" opp CSS, HTML, og Javascript-kode avhengig av hva slags plattform programmet skal kjøres på, og kan kjøres på forskjellige enheter fordi Cordova har tilgang til den native API'en til enheten. Dette fører til at de resulterende applikasjonene utviklet med Cordova blir såkalte hybrid-applikasjoner som ikke er primært mobil eller web-baserte.

Apache Cordova er en del av PhoneGap engine.

Node.js (versjon 2.14.12) er et åpent kryssplattform runtime-system for utvikling av server-side applikasjoner.

Basert på JavaScript.

Draw.io er et program som brukes til å lage diagrammer, flowcharts, og mye mer. Det er helt gratis og brukes i nettleseren din.

Gruppen har brukt denne nettsiden til å lage diagrammer.

Google Maps JavaScript API (Application Programming Interface) brukes til å lage lokasjonsbaserte applikasjoner. Ved å bruke Google Maps API kan utviklere integrere Google Maps i sine applikasjoner og nettsider, og det finnes stor rom for personalisering. Dette har vi brukt som basis for navigering og oversikt i appen.

jsfiddle (jsfiddle.net/)

Det er et online programmering program som kjøres Integrated Development Environment (ide). Her blir kan man teste javascript, html og css.

PROGRAMMERINGSSPRÅK

Herunder er en liste over de forskjellige programmeringsspråkene vi har brukt.

JavaScript

- jquery-1.7.1.min.js
- jquery.mobile-1.0.1.min.js

JavaScript er et skriptspråk som er kjent for å tilføre dynamiske elementer i nettsider. Dette kjøres lokalt i nettleseren og syntaksen er inspirert av Java. Vi har kunnet ta i bruk dette sammen med HTML5 fordi vi utvikler en kryssplattformapplikasjon ved hjelp av Cordova, som nevnt tidligere.

HTML5 (HyperText Markup Language) er et markeringsspråk som brukes til å utforme nettsider. Markeringsspråk vil si at det ikke kun er alminnelig tekst som skrives, men også instruksjoner for hvordan tekstens oppsett skal være. Altså brukes HTML for å legge inn data og strukturere nettsider.

CSS (Cascading Style Sheets) er et språk som brukes til å definere utseende til nettsider, og supplementerer HTML. Her kan man definere for eksempel bakgrunnsfarge og teksttyper inne i ett dokument, i stedet for å gjøre det i hver eneste HTML-fil. Kalles også for gjennomgående stilark på norsk.

jQuery Mobile

- jquery.mobile.structure-1.0.1.css
- jquery.mobile-1.0.1.css

jQuery Mobile er et HTML5-basert brukergrensesnittsystem som er utviklet for å lage responsive applikasjoner på mobil, nettbrett og PC. Det bruker HTML, CSS og JavaScript for å legge opp sider med så lite scripting som mulig. Basert på “write less, do more”, designer den nettsider automatisk for et simpelt utseende som er lett å bruke, og som fungerer på alle mobile enheter.

UTVIKLINGSPROSESS

Her beskriver vi hvordan prosjektutviklingsprosessen i sin helhet har foregått i løpet av semesteret. En mer detaljert beskrivelse av systemet kan finnes i produktrapporten, og her tar vi for oss kun hvordan arbeidet har foregått siden oppstart - blant annet hvilke hinder som har oppstått og hvilke endringer som måtte gjøres for å kunne levere et godt produkt.

OPPSTARTSFASEN

Etter første møte med arbeidsgiver hvor vi fikk oppgaven om å lage applikasjonen, diskuterte vi hvordan vi skulle gå frem med å begynne arbeidet på oppgaven. Vi tok for oss blant annet hva som var forventet av oss, hva vi var i stand til å fullføre ettersom vi er et mindre team, hvilken funksjonalitet som skulle prioriteres først, og hvilke verktøy som var best egnet for å lage den løsningen arbeidsgiver var ute etter.

Vi kom over Cordova som tillot oss å kode i HTML/CSS/JavaScript som om vi skulle lage en ordinær webapplikasjon, og slo fast at dette var det beste verktøyet å bruke ettersom vi ikke måtte skrive kode spesifikk for ulike enheter slik som nettbrett og PC ettersom Cordova pakker inn koden og kjører den med samme funksjonalitet på alle enheter.

Å samle informasjon om hundrevis av bedrifter og legge det i en database ville ha vært en svært tidkrevende prosess, spesielt for kun to personer. Vi regnet med at mesteparten av tiden vår ville gå til utviklingen av selve applikasjonen. Vi så derfor på Google Maps API som et godt alternativ; ved å integrere dette i applikasjonen vår fikk vi mulighet til å hente ut informasjon og lokasjoner til bedrifter uten bruk av en database, samtidig som bruksområdet til applikasjonen ble utvidet.

Etter å ha fått utviklingsverktøyene på plass startet vi å skrive styringsdokumenter for å hjelpe oss videre ved å ha konkrete referansepunkter for hva som måtte gjøres til hvilken tid. Siden vi

hadde fått stor grad av frihet fra arbeidsgiver var det viktig at vi fikk våre tanker om løsningen ned på papir og et helhetlig bilde av hvordan jobben skulle gjennomføres, med arbeidsgivers krav som retningslinjer.

Vi bestemte oss for å ha enkelhet som et utgangspunkt - hvordan skal appen se ut når man åpner den, og hvordan gjør vi det intuitivt å bruke den? Siden forsiden av appen naturligvis er det første brukeren ser, startet vi med å tegne noen skisser over hvordan den skulle se ut. Vi ville at brukerene raskest mulig skulle finne fram til det de trengte, og da var det behov for et ryddig grensesnitt som alle forstår ved første blikk. Skissene kan ses i følgende *figur 4.3*, og større versjoner av bildene kan finnes i kapittel 7 – *appendiks og vedlegg*.

Figur 4.3 – Skisser utarbeidet i oppstartsfasen

Etter hvert kom vi fram til at vi burde ha tre knapper på forsiden, med det vi tror kommer til å bli mest brukt øverst. Ved å trykke på en av disse kommer man rett inn i for eksempel å søke opp bedrifter i nærområdet.

MÅLGRUPPE

En applikasjon for å søke opp bedrifter, butikker og restauranter kommer til å appellere til et bredt publikum, så den må ikke være rettet mot en spesiell gruppe mennesker, for eksempel ungdom. Vi måtte naturligvis da ta hensyn til at blant annet eldre eller personer med nedsatt synsevne kunne bruke applikasjonen, og vi har gjort tekst og knapper store for å gjøre opplevelsen bedre.

KRAVSPESIFIKASJONEN OG ENDRINGER

Kravspesifikasjonen som vi selv utarbeidet ut i fra de veiledende punktene fra arbeidsgiver har spilt en viktig rolle og har veiledet oss fra start til slutt. Det har ført til at vi har holdt en fast kurs hele veien og vært klare over begrensningene vi måtte holde oss innenfor. Det har blitt gjort noen få endringer til kravspesifikasjonen underveis, men dette var for det meste tilknyttet ønsket funksjonalitet og hvordan brukergrensesnittet skulle se ut.

Vi droppet å implementere en knapp som oppdaterer brukerens posisjon, fordi vi ikke fikk bruk for den. Brukerens posisjon oppdateres automatisk når man åpner kartet.

Ønskede funksjoner som anmeldelser for bedrifter og knapper for deling til sosial media ble også utelatt fra den ferdige løsningen. Dette på grunn av tidsmangel, siden disse funksjonene ble nedprioritert.

OPPSUMMERING

I løpet av semesteret har gruppen lært mye nytt om applikasjonsutvikling, samarbeid og fått erfaring som vi håper på å ta med videre til arbeidslivet. Samarbeidet har fungert godt, og dette var helt nødvendig for at vi skulle lykkes med prosjektet vi satt ut på. Vi har samtidig fått erfare hvordan det er å takle tidspress og utfordringer, og det har vært en krevende men lærerik prosess.

Vi føler at vi har vært for dårlige med å planlegge hvordan prosjektet skulle gjennomføres. Dette skyldes nok at vi ikke helt forsto hvor stor oppgaven var, og at vi brukte lang tid på å finne ut hvordan vi skulle gå frem med å begynne utviklingen. Med bedre planlegging kunne vi nok fått levert et bedre produkt, men vi har forsøkt å legge opp til videreutvikling så godt vi kan.

Begge gruppemedlemmene har hatt erfaring med lignende type oppgaver, dog ikke av samme størrelse som dette prosjektet. Vi har oppdaget nye utviklingsverktøy og fått et innblikk i hvor viktig det er å ha gode arbeidsrutiner og kommunikasjon, i tillegg til at vi har fått en viss forståelse for hvordan applikasjoner bygges opp og hva slags krav som bør settes for dem. Vi kan si at vi har hatt et godt utbytte av prosjektarbeidet og det har vært en veldig lærerik periode fra oppstart i slutten av 2015 til avslutningen i mai 2016.

Del 4 - Produktrapport

FORORD

Vi forutsetter at leseren har lest den tidligere dokumentasjonen og presentasjonen av prosjektet i forkant av denne produktrapporten for å ha fått et innblikk i gruppen og utviklingsprosessen, samt en forståelse for hva prosjektet omhandler. Denne rapporten kan da leses selvstendig og her beskrives det tekniske rundt applikasjonen.

Vi har forsøkt å holde språket i denne rapporten så enkelt som mulig slik at det kan leses av alle, men det vil likevel dukke opp ord og begreper som kan være forvirrende hvis leseren ikke har en oppdatert datateknisk forståelse. Vi henviser til ordlisten som kan finnes i appendiksen (kapittel 7) dersom noe ikke skulle være klart.

KORT OM PROGRAMMET

Applikasjonen har som formål å gi informasjon om ulike butikker og spisesteder i Norge. Brukeren skal kunne søke opp butikker og få informasjon om åpningstider og kontaktinfo, samt lokasjonen til butikken og en veibeskrivelse. Hensikten er at brukere skal ha en lettvint måte å finne butikkene de er ute etter.

BYGGEKLOSSER

Applikasjonen er utviklet ved hjelp av rammeverket Apache Cordova, og bruker Google Maps API for å hente fram lokasjoner og informasjon. HTML5, CSS og JavaScript er språkene som er brukt i utviklingen av løsningen.

GRENSESNITT

Her gjør vi rede for hvordan applikasjonen fremstilles for brukeren. Her tok vi utgangspunkt i skissene våre som igjen var basert på kravspesifikasjonens punkter om at applikasjonen skal være enkel å bruke og ha et minimalistisk utseende. Vi fokuserte mest på funksjonalitet og hvor knapper skulle plasseres i første omgang, i stedet for å hoppe rett på det visuelle. Ting som fargevalg og skrifttyper ble satt til side i starten av utformingen til brukergrensesnittet.

SKISSER

Skissene ble tegnet i startfasen av prosjektet, når vi trengte et veiledende visuelt grunnlag som basis for å kunne begynne å utarbeide det endelige grensesnittet. Vi så ikke på disse som et

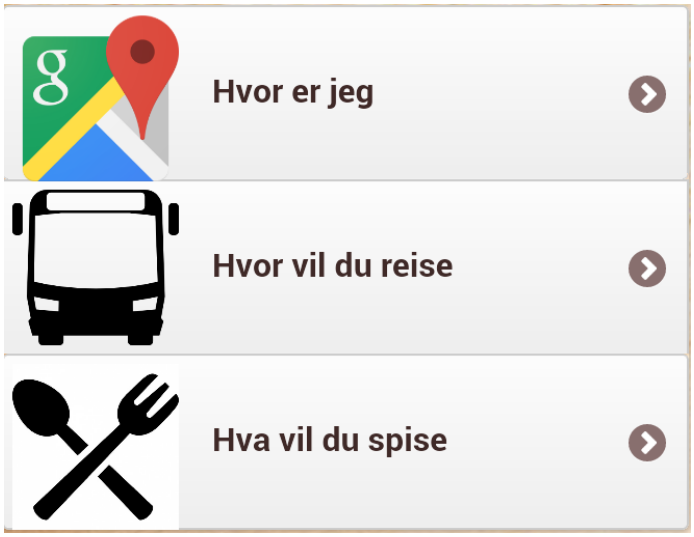
absolutt krav, men heller en pekepinn på hvordan oppsettet bør se ut. Disse kan finnes i prosessrapporten under *oppstartsfase* eller som *vedlegg 1* i kapittel 7.

GRENSESNITTET

Arbeidet på brukergrensesnitt ble startet med å plassere knapper og tekstbokser. Vi brukte HTML5 og CSS for designet, og denne koden ble pakket inn av Apache Cordova som gjorde det mulig å kjøre koden på ulike plattformer. Vi har lagt med noen bilder av det ferdig utviklede brukergrensesnittet som vedlegg i kapittel 7, og dette er løst basert på skissene som vi lagde.

KNAPPER OG IKONER

For at funksjonaliteten i programmet skulle være tydelig gjorde vi knapper store og tilføyde ikoner og tydelig tekst til dem for at det skulle være enklest mulig for brukeren å forstå hva applikasjonen gjør (fig. 4.1).



Figur 4.1 – Knappene i det endelige grensesnittet.

TEKSTBOKSER

Tekstboksene har tydelige mørke rammer for å skille dem ut fra andre elementer og bytter til en blåfarge når de aktiveres som en tilbakemelding til brukeren. Her er automatisk oppslag støttet slik at man ikke trenger å skrive inn et helt navn eller adresse for å gjøre et søk (se figur 4.2).

Figur 4.2 – Søk i tekstboks

Her er det også mulig å søke på ulike typer mat, i stedet for kun butikker som i bildet over. Man kan for eksempel taste inn «burger», og alle spisesteder i nærheten som serverer det vil dukke opp på kartet.

Man må imidlertid slette en bokstav av gangen hvis man har skrevet feil eller ombestemmer seg; meningen var at vi skulle ha en knapp som slettet hele tekststrengen for å unngå dette, men vi oppdaget ikke dette før det var for sent og vi ikke hadde tid til å implementere det.

LYD

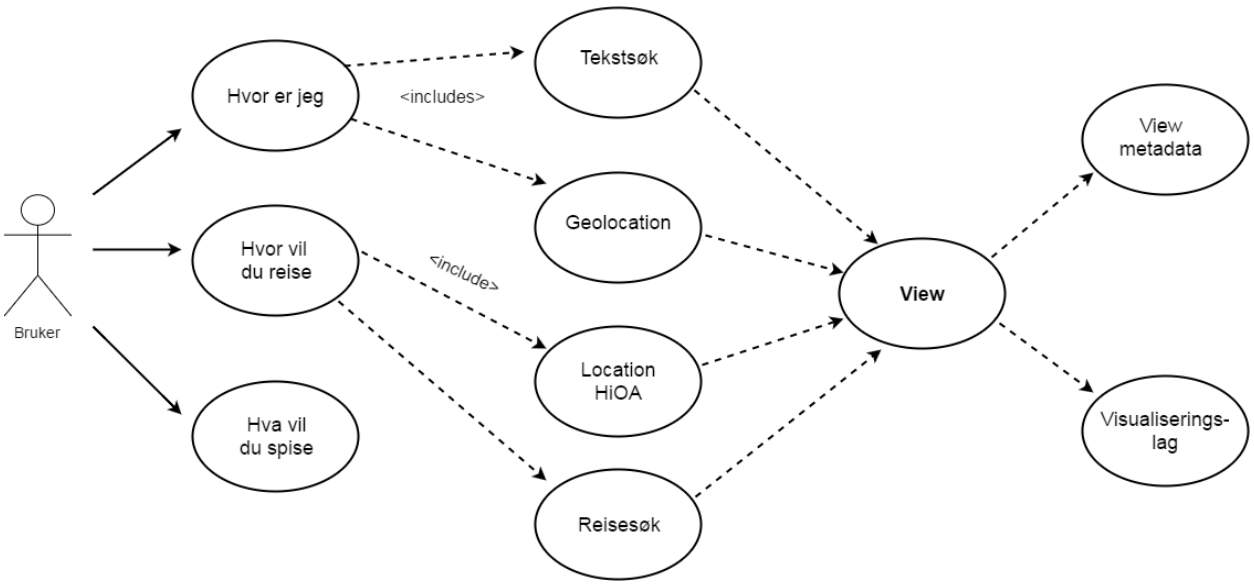
Applikasjonen inneholder ingen lyd. Dette fant vi ikke noe praktisk bruk for og så derfor ikke på det som en nødvendighet. Hvis lyd skulle inkluderes måtte det implementeres alternativer for kontroll av lyden, noe som tar opp en god del ressurser og senker graden av universell utforming hvis vi av en grunn ikke skulle få det til.

FARGEVALG

Vi hadde ikke noe konkret utgangspunkt for fargevalg, og valgte det vi synes var stilrent. Vi ville holde det enkelt og begrenset antall farger så mye vi kunne. I hovedsak består grensesnittet av svarte, hvite, og grå farger med et lyst utseende. Fargevalget er også tilrettelagt fargeblinde som bruker applikasjonen slik at de ikke opplever noen vanskeligheter.

NAVIGERING

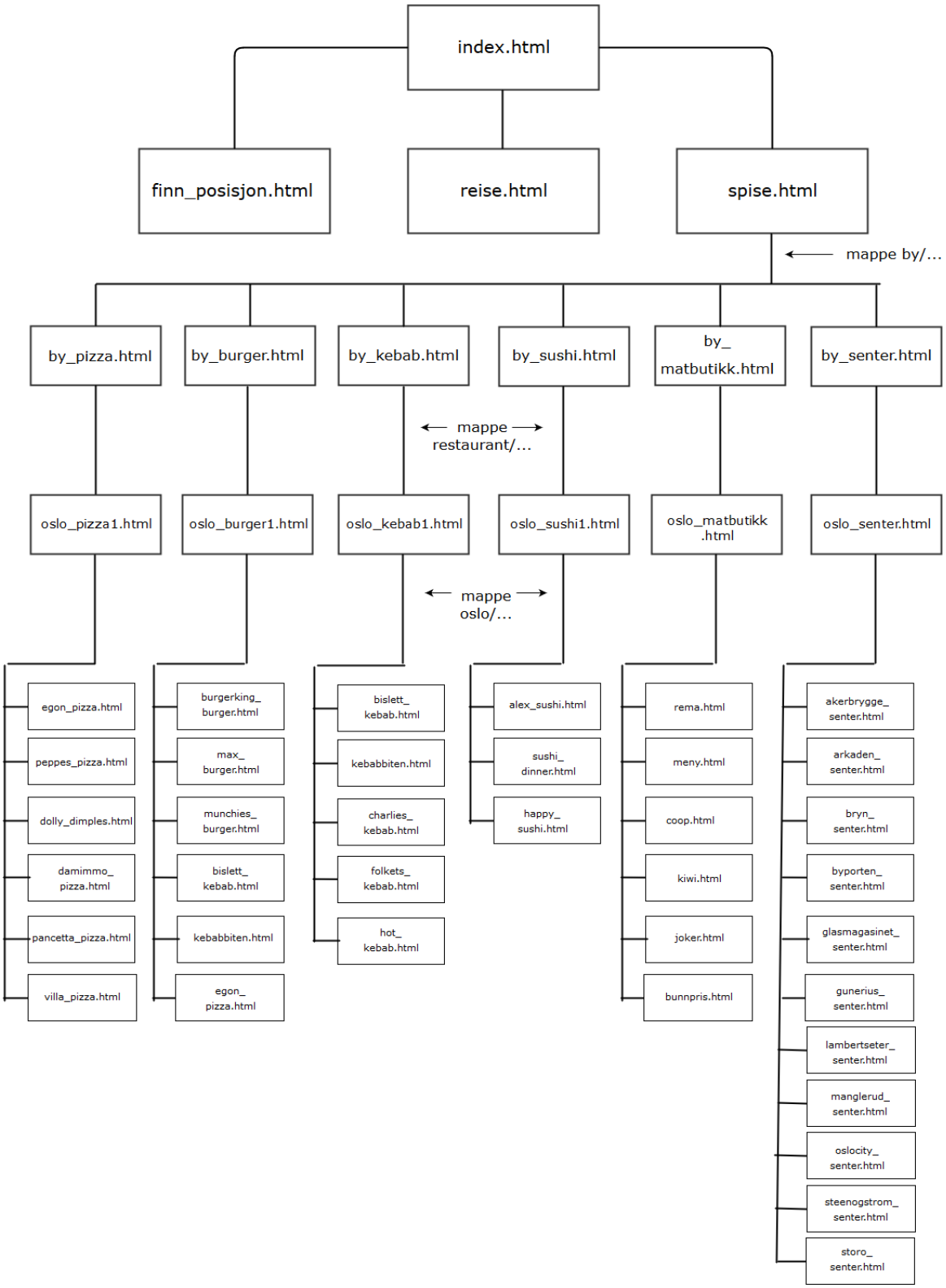
Å finne fram i appen skal være en rask og enkelt forståelig prosess. Vi lagde derfor tre knapper på forsiden (som sett på figur 4.1), med det vi tror er mest interessant for brukeren øverst. Ved å trykke på disse hopper man rett inn i et kart og trenger bare å taste inn nødvendig informasjon i tekstboksene. Den siste knappen, «hva vil du spise», inneholder ikke et kart – men en oversikt over ulike typer mat eller butikker som finnes i en gitt by. Herfra kan man for eksempel velge kategorien «Kebab», byen «Oslo», og en liste over kebabsjapper vil vises.



Figur 4.3 - Use Case diagram.

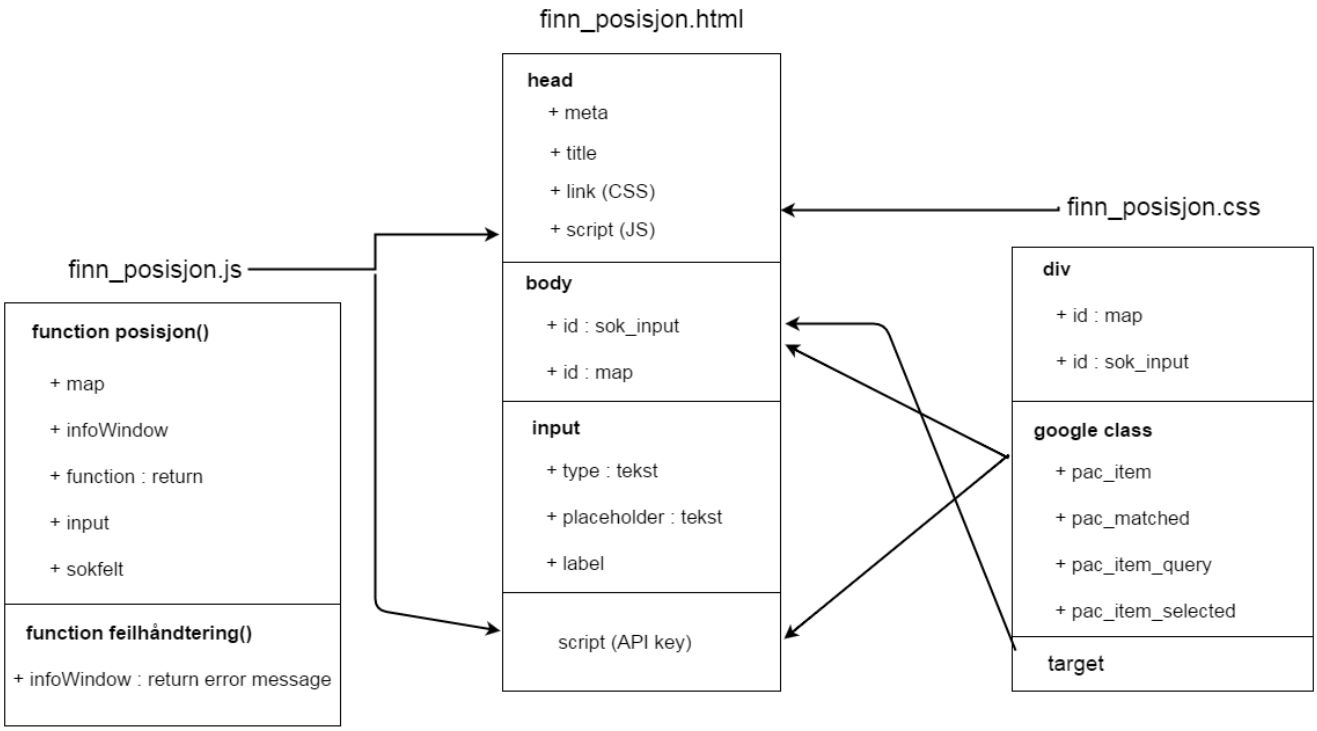
APPENS OPPBYGGING

Her viser vi hvordan de ulike delene av systemet er bygget opp og strukturert.

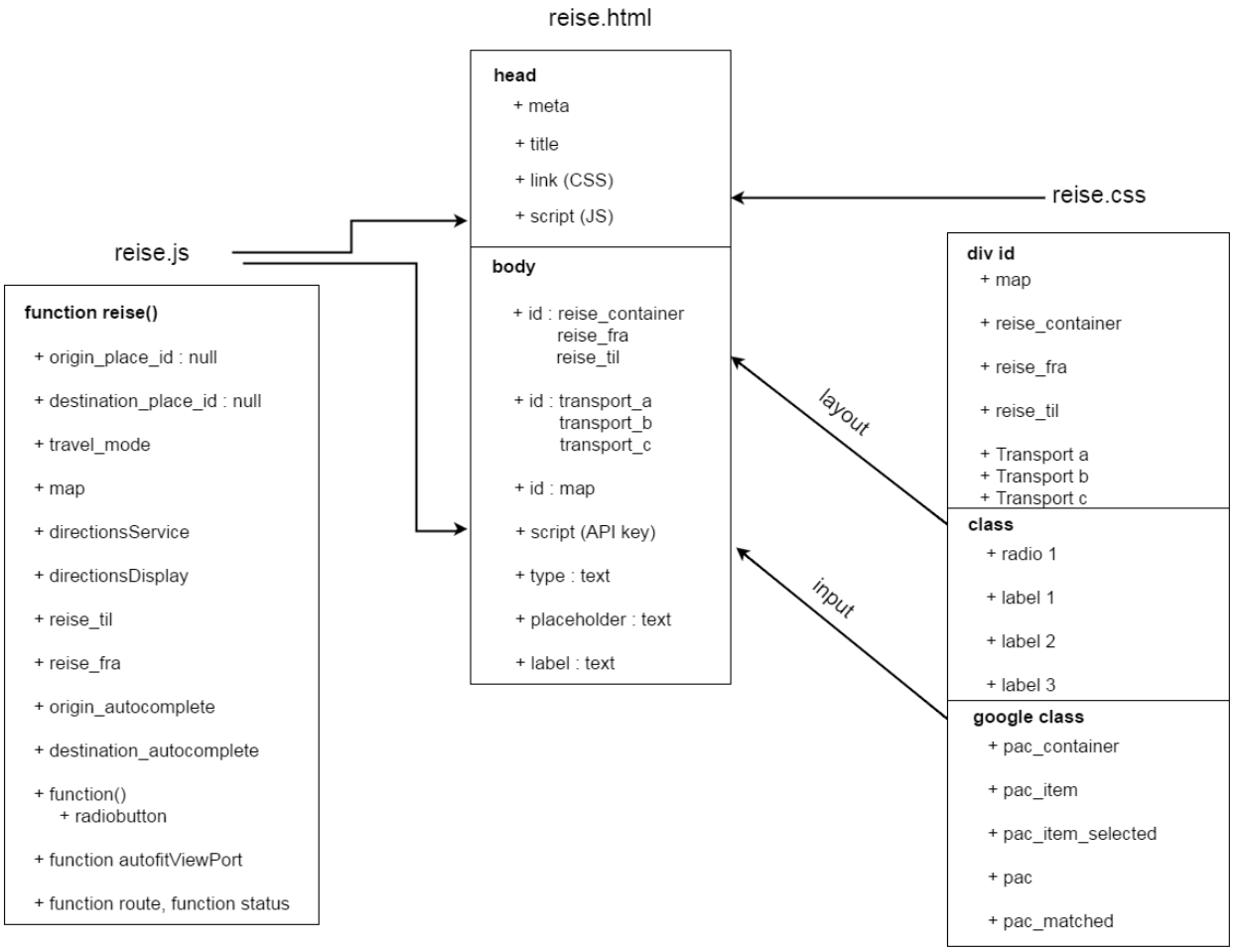


Figur 4.4 – Oppsettet av HTML-filene

Figuren over viser hvordan HTML-sidene er lagt opp. Vi starter naturligvis med *index.html*, som inneholder *finn_posisjon.html (fig. 4.5), reise.html (fig. 4.6), og spise.html*. Av disse er det kun *spise.html* som har undermapper. Her befinner listen over spisesteder seg (i tillegg til matbutikker og kjøpesentre). Etter å ha valgt en kategori i *spise.html*, kommer man til mappen *by*, som er en liste over norske byer. Her må man velge en by før man kan fortsette. Herifra kommer man til mappen *restaurant*, og en liste over spisesteder i byen du har valgt vises.



Figur 4.5 – Strukturen til finn_posisjon.html



Figur 4.6 – Strukturen til reise.html

VALG AV FUNKSJONER

Det starte med at vi gikk på nettet leste mye hvordan oppbygging av hybrid app til alle platform. Vi leste litt og teste par programmer på starten som Eclipse, Visual studio 2015 og netbeans. Oppsett til Eclipse funke ikke som vi ville, dermed prøvd vi på visual studio 2015 men vi var ikke så fornøyd med det siden vs 2015 levere bing map og ikke google api. Dermed valgt vi å kjører på med netbeans 8.

Da var det å teste om vi skulle bruke den nyeste version av netbeans 8.1 eller 8. Vi prøvdet første med gamle netbeans 8 som ikke hadde alt vi behøvde. Da gikk vi over til netbeans 8.1 det virka som vi ville.

Installasjonen var veldig komplisert ved å legge in Node.js, , Git, Android bundle sdk, Windows phone sdk, Ant-build .

Design

Vi starte med å lage en app med mange knapper, som end med 3 knapper på starten av app.

Bakgrunnen har vi prøvd mange forskjellig bilder og bakgrunn med tanken på at det skal se frisk og pent.

Vi har brukt mest tid på design våre program til app og har ikke brukt mye tid på hvordan det skal se ut på hjemmesider. Løsning vi har kommet til er at vi skape en ny css fil som kjøres til hjemmesider og en til mobilt. Dette vil ikke påvirke i noe som helt i programmet enn å bytte import på header i html. Men dette har vi da ikke fått tid til.

Finn_posisjon og reise har vi bare stort sett lest dokumentasjon til google maps javascript på deres hjemmesider og prøvd oss fram, det vi synes er best og enkelt bruke. Test verktøy vi har mest bruk her er jsfiddle.net for å sjekke og test hvordan Google Maps fungere og implementere inn på vår program senere.

Spise.html har vi bare brukt vår kunnskap gjennom det vi har lært og brukt ulike hjemmesider som jquerymobile.com og w3schools sider for å settes inn.

Valg av ulike bibliotek

Vi har valgt gå for Jquery.mobile.structure-1.0.1.css og Jquery.mobile.1.0.1.css siden det funke bra til hybrid app og veldig enkelt å bruke. Selv om det tok litt tid for å skjønner hvordan jquery-mobil fungere og hvor alt skal ligge.

Vi har også tenkt kanskje legge inn andre design ferdig css som bootstrap, men vi er fornøyd med jquery-mobile sin.

Struktur på program

Vi har også lagt veldig struktur på ting. Der alle css, js, html file skal ligge på en egen mappe, navn dem samme med bare forskjellig syntaks, så det blir lett å finne fram i store mengder av filer. Der de viktigste filen skal ligge utenfor noe mapper så de kan bare hentes lett inn.

Valg av plugins

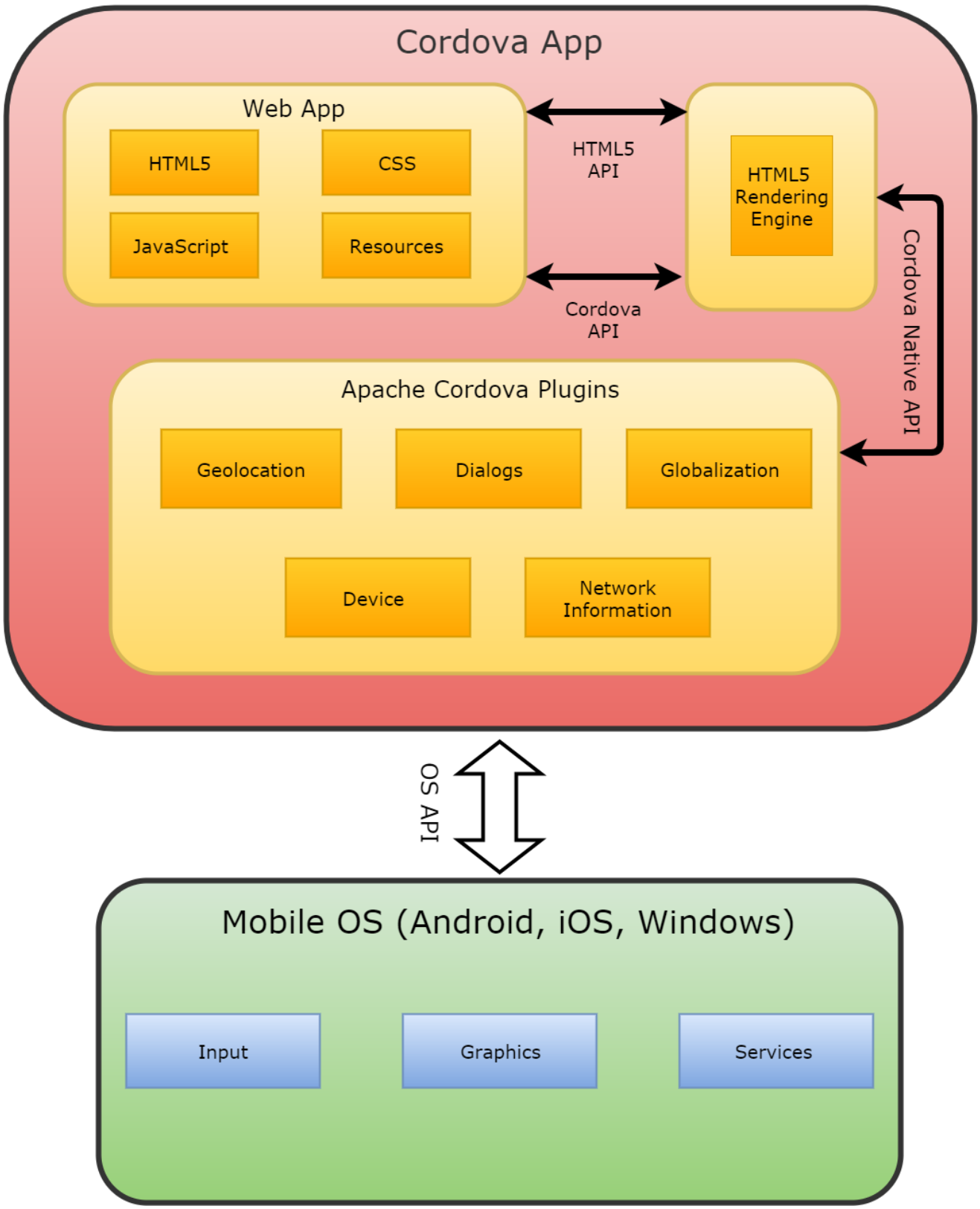
Vi har bare valgt de viktigste cordova plugins som skal brukes til appen, så den kan kjøres Kartet og koble seg til internett.

TEKNISK BESKRIVELSE AV APPEN

I denne delen beskriver vi generelle tekniske aspekter og funksjoner i applikasjonen.

APACHE CORDOVA

Det er nevnt tidligere i prosessrapporten at vi har benyttet oss av Cordova, som er rammeverket vi har brukt i utviklingen. Dette er et godt verktøy fordi utviklere av mobile applikasjoner kan forlenge applikasjonen over mer enn én plattform uten å måtte implementere den på nytt med hver plattforms språk og verktøysett. Man kan tenke på Cordova som en container som tillater en webapplikasjon til å ta i bruk mobile funksjonaliteter (for eksempel kamera og geolokasjon). Webapplikasjoner kan ikke ta i bruk disse funksjonalitetene av seg selv, så det er her Cordova kommer inn og fungerer som en bro som kobler webapp til mobilenhet. Virkemåten kan ses nærmere i figur 4.7.



Figur 4.7 – Arkitekturen bak Cordova-applikasjonen

WebView (HTML5 Rendering Engine)

Den Cordova-aktiverte *WebView'*en utgjør hele brukergrensesnittet til applikasjonen.

Web App

Dette er delen hvor all koden for applikasjonen befinner seg. Selve applikasjonen er implementert som en nettside, i en lokal fil *index.html* som peker til JavaScript, CSS, bilder og andre ressurser som kreves for at den skal kunne kjøre. Applikasjonen kjøres i en *WebView* innenfor den native applikasjons-*wrapperen*.

Programmets *config.xml* er en viktig fil som gir informasjon om applikasjonen og spesifiserer parametere som påvirker hvordan den fungerer. Hvis du for eksempel vil at applikasjonen din skal reagere på orienteringen på en mobil og bytte til landskapsmodus, spesifiseres det her.

Cordova Plugins

I illustrasjonen over ser man at en Cordova-applikasjon bruker flere plugins. Disse er kodepakker som tillater Cordova til å kommunisere med plattformen applikasjonen kjøres på gjennom JavaScript, og er helt essensielle for systemet. Det finnes såkalte *kjerne*-plugins, som gir tilgang til mobilfunksjoner som batteri og kamera, og det er mulig å lage sine egne eller installere tredjeparts-plugins. Sistnevnte kan finnes på https://cordova.apache.org/plugins/, og siden har i skrivende stund over tusen hjemmelagde kodepakker på lager.

Når man oppretter et Cordova-prosjekt vil ikke noen plugins være til stede, selv ikke kjerne-plugins. Disse må man legge inn selv. De vi har brukt til dette prosjektet er følgende:

(ref: cordova)

Geolocation (`cordova-plugin-geolocation`)

Gir informasjon om enhetens plassering, slik som bredde- og lengdegrad.

Denne pluginen definerer et globalt `navigator.geolocation`-objekt.

Dialogs (`cordova-plugin-dialogs`)

Gir tilgang til native brukergrensesnittelementer, slik som dialogbokser, gjennom et globalt `navigator.notification`-objekt.

Globalization (`cordova-plugin-globalization`)

Henter informasjon om brukerens lokalisering, tidssone og språk.

Denne pluginen definerer et globalt `navigator.globalization`-objekt.

Device (`cordova-plugin-device`)

Henter beskrivelse av enhetens maskin-og programvare gjennom et globalt `device`-objekt.

Network Information (`cordova-plugin-network-information`)

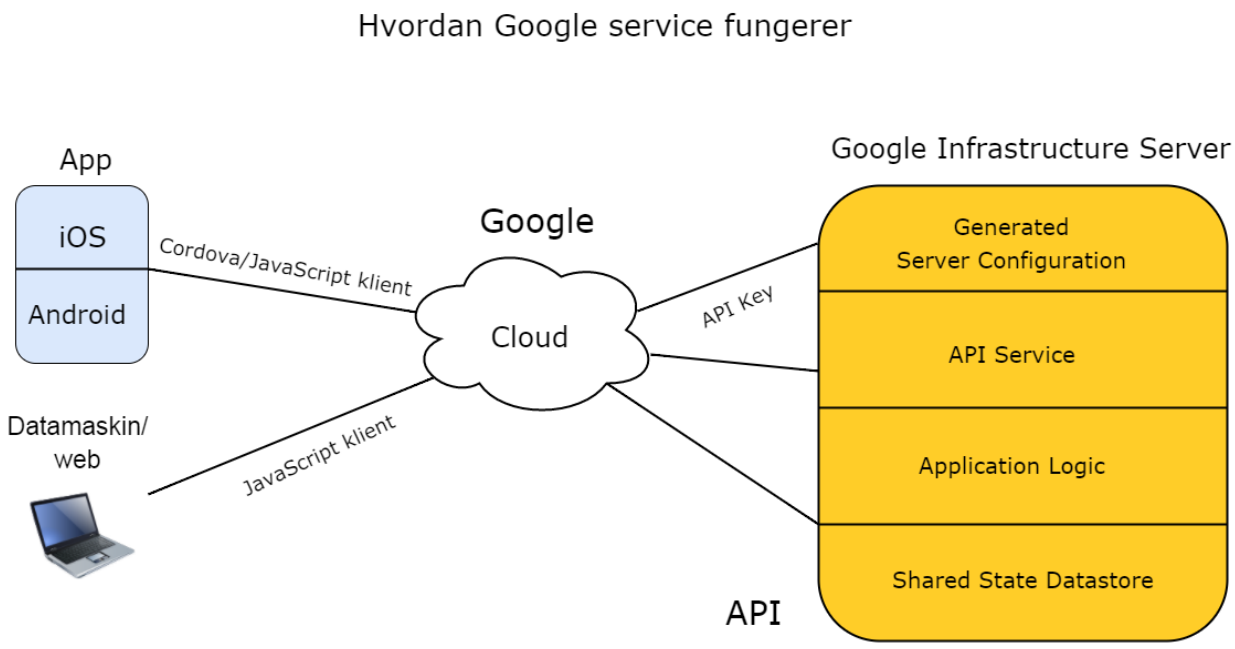
Henter informasjon om enhetens mobilnett og WiFi-tilkobling, og om enheten er tilkoblet internett, gjennom et `connection`-objekt.

GOOGLE MAPS API

Vi har integrert Google Maps API med Cordova for å skape et interaktivt kart. Det er et kraftig verktøy som tilbyr en rekke tjenester for datavisualisering, manipulasjon av kart, veibeskrivelse og mer.

API-en krever at man oppretter en token/API key som skal brukes med applikasjonen. Dette er et sikkerhetstiltak for å garantere at det er kun vår applikasjon som har tilgang til vårt API.

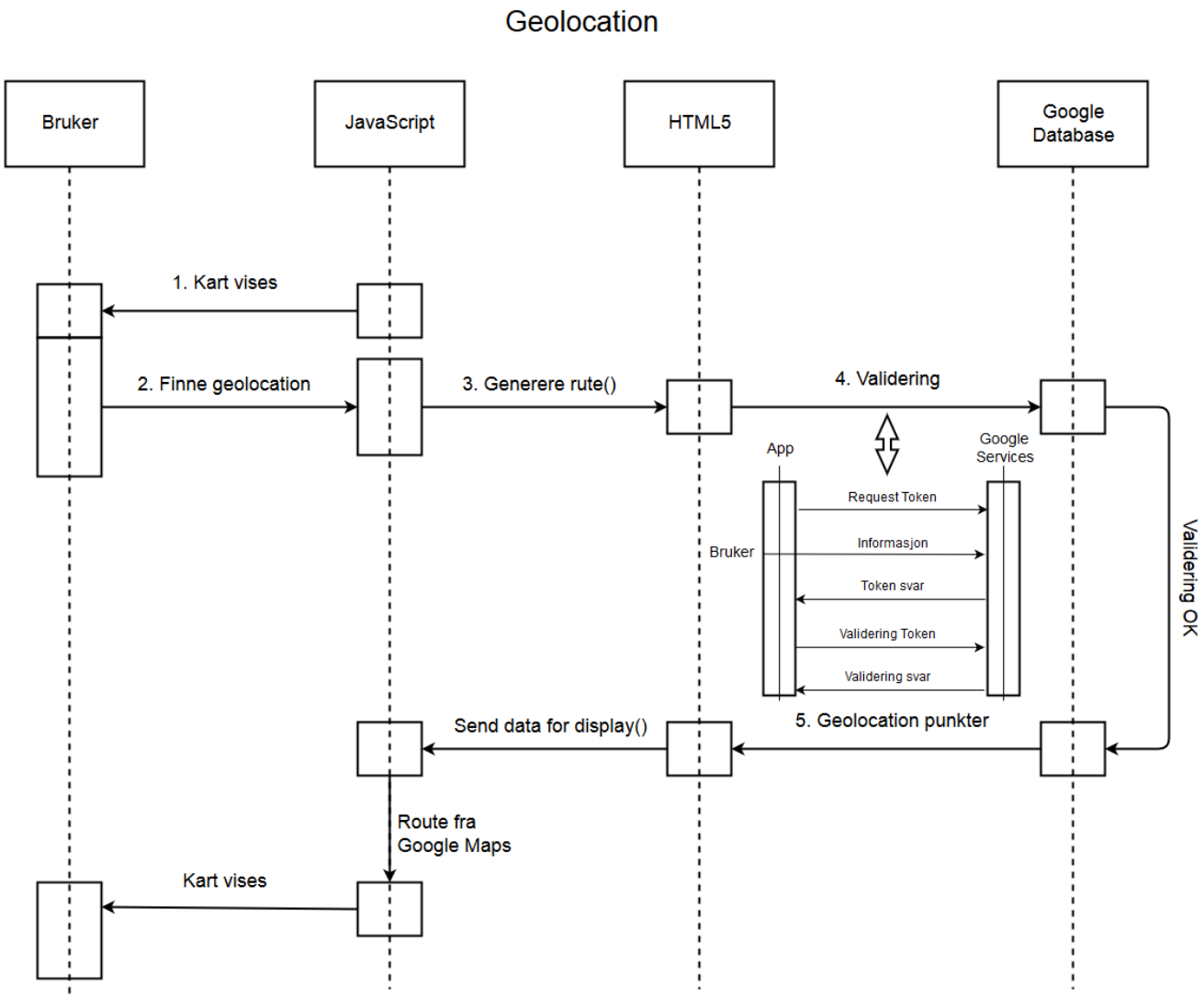
Det negativt med dette er at Google tar betalt når man har mange som bruker produktet. De har bare sagt på sine hjemmesider at hvis det overstiger mer enn 250.000 bruker pr dag. Da vil Google kontakte deg om betaling på service.



Figur 4.8 – Hvordan Google service fungerer

GEOLOCATIONS SEKVENSDIAGRAM

Den beskrive hvordan bruker får kartet sitt automatisk på appen når finn_posisjon knappen blir trykk på. Vi har da ikke tatt med selveste metode navnet som get og set, men heller skrive hvordan det skjer på en språk alle kan skjønne. Noe av uttrykke her er engelsk, for det kan ikke oversett helt riktig til norsk.



Figur 4.9 – Geolokasjon

PROGRAMKODE OG BESKRIVELSER

FINN_POSISJON.HTML

```
<html>
    <head>
       <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
        <meta charset="utf-8">

        <title>Finn din Posisjon</title>

        <link rel="stylesheet" href="css/finn_posisjon.css" />
        <script src="js/finn_posisjon.js"></script>

    </head>

    <body>
       <input id="sok_input" class="sok_control" type="text" placeholder="søk et a
    <div id="map"></div>

     <script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCnCBIKiSfSbJ-2
         async defer></script>

    </body>
</html>
```

Filen starter med metadata som definerer hvordan siden skal se ut i nettlesere. Her spesifiseres det at brukeren ikke kan forandre på størrelsen og at det skal tilpasses til full størrelse på skjermen til mobile enheter.

```
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
```

Title viser navn på siden i nettlesere.

```
<title>Finn din Posisjon</title>
```

Link brukes til å definere en link mellom et dokument og en ekstern ressurs. Her er den brukt til å linke til et eksternt stilark (CSS-fil).

Script definerer et JavaScript-skript på klientsiden. Her peker den til en ekstern skriptfil.

```
<link rel="stylesheet" href="css/finn_posisjon.css" />

<script src="js/finn_posisjon.js"></script>
```

På body har vi hvordan siden skal se ut innholdsmessig. Her blir det lagt inn kart som skal vises og en ID som spesifiserer objektet.

Her prøver vi å verifisere token eller API key som trengs for å kunne kjøre kartet fra Google Maps API.

```
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCnCBIKiSfSbJ-
2NQfF3Syql6wdgveG-L0&libraries=places&callback=posisjon" async defer></script>
```

FINN_POSISJON.JS

Her kjører vi gjennom en funksjon for posisjon og en funksjon for feilhåndtering av kartet.

Det første vi ser i skriptet er at det hentes kart fra Google Maps med geolokasjonen til Høgskolen i Oslo og Akershus som startpunkt.

```
var map = new google.maps.Map(document.getElementById('map'),

        {

            //geolocation adress til hioa. startpunkt.

          center: {lat: 59.9295700, lng: 10.7355620},

          zoom: 19

        });
```

Deretter kjøres det en oppdatering av kartet som går gjennom `navigator.geolocation` og automatisk finner din posisjon med breddegrad og lengdegrad. Det sendes etterpå en melding av hvor du er på kartet og viser deretter din lokasjon midt på kartet.

Så følger en if-statement for feilhåndtering av kartet når den ikke klarer å dirigere kartet sentralt. Det hentes fram en feilmelding hvis det oppstår en feil.

```
if (navigator.geolocation)

         {

             navigator.geolocation.getCurrentPosition(function(position)

             {

                    var pos =

                            {

                                lat: position.coords.latitude,

                                lng: position.coords.longitude

                            };
```

```
                infoWindow.setPosition(pos);
                infoWindow.setContent(' <strong>Du er her');

                 map.setCenter(pos);
             }
                  ,

             //feilhåndtering. hvis map ikke kan oppdatere seg.
            function()
            {
                     handleLocationError(true, infoWindow, map.getCenter());
            });
        }
        else
            {
                // feilmelding for håndtering av browser
                handleLocationError(false, infoWindow, map.getCenter());
            }

function handleLocationError(browserHasGeolocation, infoWindow, pos)
      {
        infoWindow.setPosition(pos);
        infoWindow.setContent(browserHasGeolocation ?
                        'Error: Geolocation service funker ikke' :
                        'Error: Geolocation støtter ikke denne browser');
      }
```

Her vil vi legge til funksjoner som kjøres til kartet med input og henter fram det man søker til kartet.

Når man foretar et nytt søk, vil de gamle markørene på kartet fjernes og erstattes med resultatene fra det nye søket.

```
map.addListener('bounds_changed', function()
        {
          sokfelt.setBounds(map.getBounds());
    });


        var markers = [];


         sokfelt.addListener('places_changed', function()
        {
          var places = sokfelt.getPlaces();


          if (places.length == 0)
          {
                return;
          }


          markers.forEach(function(marker)
          {
                marker.setMap(null);
          });
```

```
markers = [];
```

Her vil vi lage en markør for hvert enkelt sted som skal vises på kartet når man søker etter et gitt navn.

```
places.forEach(function(place)
{
      var icon =
              {
                  size: new google.maps.Size(71, 71),
                  origin: new google.maps.Point(0, 0),
                  anchor: new google.maps.Point(17, 34),
                  scaledSize: new google.maps.Size(25, 25)
              };

      markers.push(new google.maps.Marker(
       {
              map: map,
              icon: icon,
              title: place.name,
              position: place.geometry.location

      }));
```

REISE.HTML

Denne filen er svært lik *finn_posisjon.html* som vi har forklart hvordan fungerer. Derfor tar vi ikke med forklaring på den.

REISE.JS

Vi starter med det første å definere og deklarere alle objekter for å kjøre funksjonen til kartet.

```
var reise_til_id = null;
        var reise_fra_id = null;
        var travel_mode = google.maps.TravelMode.WALKING;


        var map = new google.maps.Map(document.getElementById('map'),
        {
           mapTypeControl: false,
           center: {lat: 59.9295700, lng: 10.7355620},
           zoom: 17
        });


        var directionsService = new google.maps.DirectionsService;
        var directionsDisplay = new google.maps.DirectionsRenderer;


           directionsDisplay.setMap(map);
```

```
var reise_til = document.getElementById('reise_til');
var reise_fra = document.getElementById('reise_fra');

var til_autocomplete = new google.maps.places.Autocomplete(reise_til);
til_autocomplete.bindTo('bounds', map);

var fra_autocomplete =
    new google.maps.places.Autocomplete(reise_fra);
```

Her initialiserer vi en lytterknapp (radioknapp) for å velge buss, bil, eller gange.

```
function setupClickListener(id, mode)
{
   var radioButton = document.getElementById(id);
   radioButton.addEventListener('click', function ()
   {
      travel_mode = mode;
   });
}
setupClickListener('walking', google.maps.TravelMode.WALKING);
setupClickListener('buss', google.maps.TravelMode.TRANSIT);
setupClickListener('bil', google.maps.TravelMode.DRIVING);
```

Følgende funksjon brukes til å stille på plass kartet automatisk. Hvis det ikke funker vil den gå til å sette kartet til 16.

```
function autoFitViewport(map, place)
        {
           if (place.geometry.viewport)
           {
              map.fitBounds(place.geometry.viewport);
           }
           else
           {
              map.setCenter(place.geometry.location);
              map.setZoom(16);
           }
        }
```

Her vil funksjonen finne adressen du har valgt og oppdatere til posisjonen din. Hvis du skriver en ugyldig adresse vil den gi en feilmelding tilbake.

```
til_autocomplete.addListener('place_changed', function ()
            {
              var place = til_autocomplete.getPlace();
               if (!place.geometry)
               {
                  window.alert("Finner ikke location data");
                  return;
               }
               autoFitViewport(map, place);
               reise_til_id = place.place_id;
               route(reise_til_id, reise_fra_id, travel_mode,
                     directionsService, directionsDisplay);
            });
            fra_autocomplete.addListener('place_changed', function () {
               var place = fra_autocomplete.getPlace();
               if (!place.geometry) {
                  window.alert("Finner ikke noe ved denne adressen");
                  return;
               }
               autoFitViewport(map, place);
              reise_fra_id = place.place_id;
               route(reise_til_id, reise_fra_id, travel_mode,
                     directionsService, directionsDisplay);
            });
```

Denne funksjonen finner adressen det søkes til og fra. Den vil da returnere feil hvis noen ting ikke stemmer eller hvis adressen ikke blir funnet.

```
      function (response, status)
        {
                if (status === google.maps.DirectionsStatus.OK)
                {
                    directionsDisplay.setDirections(response);
                }
                else
                {
                    window.alert('Det er noe noe feil ' + status);
                }
         });
    }
```

SPISE.HTML

Vi går ikke gjennom dette, siden alt av koden er forklart nedenfor og at struktur på sidene er nesten det samme. For å unngå gjentagelser.

Del 5 - Testrapport

I denne rapporten vil vi ta for oss testingen av applikasjonen. Dette inkluderer hvordan prosessen har foregått, hvilke enheter som har blitt benyttet, og resultater fra testfasen.

HENSIKTEN VED TESTING

Det vi ønsker å oppnå med denne testrapporten er å få et innblikk i hvordan typiske brukere opplever å bruke applikasjonen vår og fjerne eventuelle feil. Siden vi også som utviklere er vant med hvordan brukergrensesnittet ser ut og fungerer, håper vi på at vi kan få tilbakemelding på det visuelle uttrykket og brukervennligheten for å deretter kunne optimalisere designet av applikasjonen hvis tiden tillater det.

ENHETER OG NETTLESERE BRUKT

Nettlesere:

- Internet Explorer/Microsoft Edge
- Mozilla Firefox
- Google Chrome

Enheter:

- PC (Ikke Mac)
- Android-telefoner 4.4.2 KitKat (HTC One X, Samsung S3)

BRUKERTESTING

Vi har testet et utvalg av 8 personer, alle jevnaldrende medstudenter ved HiOA. De fikk beskjed om å ta en full gjennomgang av applikasjonen, altså å benytte seg av alle funksjonene den har å tilby.

Ut i fra testingen kom vi fram til at applikasjonen var lett å bruke for alle.

TILBAKEMELDINGER PÅ BRUKERGRENSESNITT

Vi fikk noe kritikk for brukergrensesnittet: rammene i tekstboksene var for tykke (primært et skaleringsproblem når appen kjøres på mobil) og bakgrunnsbildet blir delt opp i fire biter på HD-skjermer (på nettlesere, men ser fint ut på mobil). Flere mente at logoen skilte seg litt for mye ut fra resten av designet med skarpheten og de sterke fargene, og burde ha mørkere toner.

TILBAKEMELDINGER PÅ FUNKSJONALITET

Det største problemet testpersonene kom over var at man i «Hvor vil du reise»-delen av appen måtte velge reisemetode før inntasting av stedsnavn, ellers fungerte ikke denne funksjonen som den skulle. Dette er en bug som vi brukte mye tid på å forsøke å løse, men kom ingen vei med. Testpersonene ville også gjerne ha mulighet til å velge sin nåværende posisjon som startpunkt, i stedet for å måtte skrive inn en adresse eller et sted.

Sånn som appen er nå, vil relevante butikker fra et søk dukke opp på kartet med en rød markør. Man må derimot zoome helt inn og trykke på ikonet for å vise informasjon om butikken eller

spisestedet. Brukerne ville at man skulle få denne informasjonen når man trykker på de røde markørene i stedet for å måtte zoome helt inn. Dette var i utgangspunktet det vi hadde planlagt skulle skje, men fikk det ikke til å fungere. Etter å ha brukt en del tid på å prøve å fikse problemet måtte vi legge om prioriteringer våre til andre ting.

Enda et ønske var muligheten til å slette tekst i søkeboksen med ett trykk. Slik det er nå må man bare bruke tastaturet for å slette bokstaver, og det kan være en plage på mobile enheter.

Applikasjonen kjører som den skal på alle plattformer og nettlesere listet ovenfor, men fungerer kun på nyere versjoner av Internet Explorer eller Microsoft Edge.

KONKLUSJON

Brukertesten har gitt oss et godt overblikk over hva som fungerer bra og dårlig med applikasjonen. Alle personene som testet appen vår opplevde at den var lett å forstå og bruke, noe som var hovedfokuset vårt i utviklingsprosessen. Feilene som ble funnet fikk vi dessverre ikke tid til å rette opp før leveranse, men testfasen var likevel viktig for å være klar over hva som må gjøres for å forbedre produktet.

Del 6 - Brukerveiledning

Denne delen beskriver hva applikasjonen gjør og hvordan brukeren skal gå frem for å oppnå sine mål.

I alle følgende skjermbilder er applikasjonen kjørt på en Android-telefon (HTC One X, Android versjon 4.2.2).

FORSIDEN

Dette er forsiden som vises når du starter applikasjonen.

Her får du tre valg:

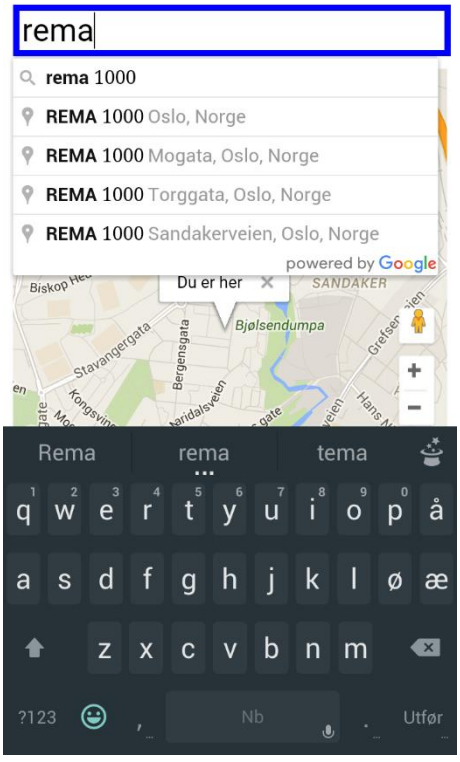
- «Hvor er jeg»: Vis din posisjon og søk opp steder i nærheten
- «Hvor vil du reise»: Få veibeskrivelse til en lokasjon
- «Hva vil du spise»: Søk i en liste over spisesteder/butikker i en spesifikk by

«HVOR ER JEG»

Når du trykker på det første alternativet vil et kart dukke opp på skjermen, sammen med en søkbar tekstboks øverst. Din posisjon vil automatisk bli oppdatert og vises sentralt på kartet.

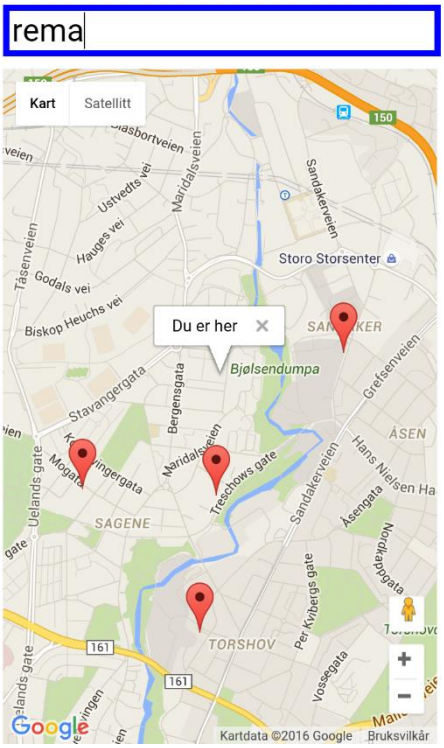
Her kan du skrive inn navnet på et spisested, en butikk, osv. for å finne steder i nærheten av der du befinner deg.

Eksempel: søk etter Rema 1000-butikker.

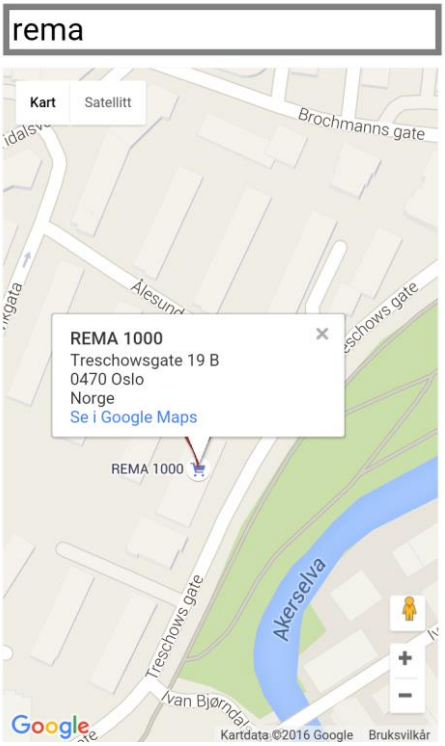


Her er det implementert en autofullfør-funksjon som viser forslag til søket ditt i en liste. Bare trykk en gang på et av alternativene for å søke. Ved å velge Rema 1000-butikken i f.eks. Torggata, vil kun denne butikken vises på kartet.

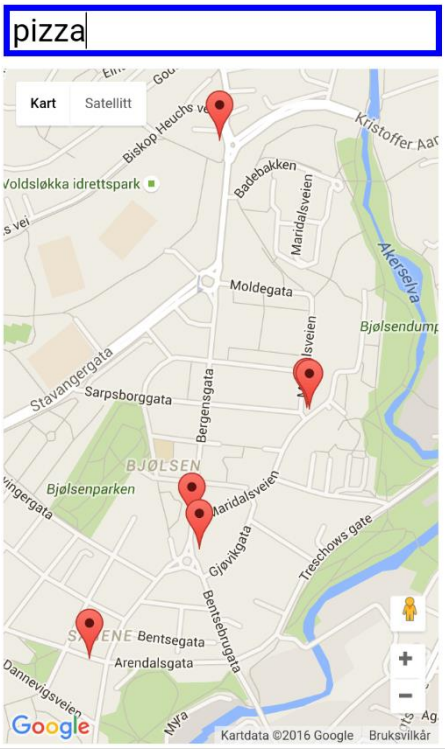
Ved å velge det øverste alternativet «Rema 1000», vil derimot alle Rema-butikker i nærheten av din posisjon vises. Her vises de relevante lokasjonene med en rød markør.



For å finne mer informasjon om en butikk, zoom inn på en av de merkede lokasjonene og trykk på ikonet til butikken.



Det er også mulig å søke på andre ting enn kun navn på butikker og spisesteder. Hvis du har lyst på pizza, kan du skrive det inn i tekstboksen og alle pizzasteder i nærheten vil dukke opp på samme måte som vist i bildene ovenfor.

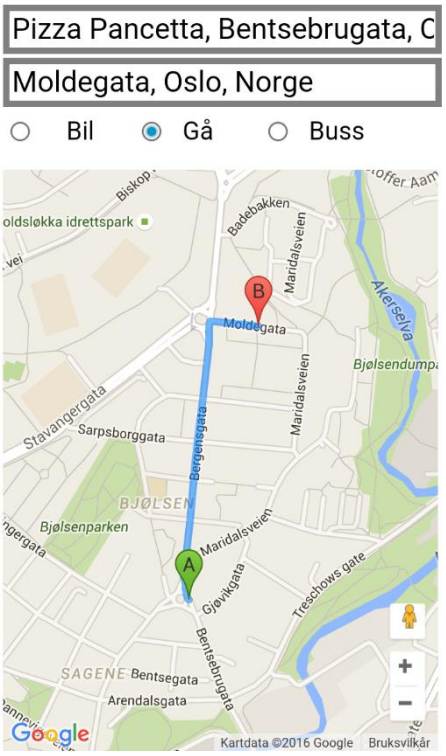


«HVOR VIL DU REISE»

Ved å velge denne funksjonen fra forsiden får du en veibeskrivelse til ønsket lokasjon ved bil, buss eller gange.

NB: Denne funksjonen har derimot en bug som gjør at du må velge reisemetode før tekst blir lagt inn, ellers fungerer ikke dette som det skal.

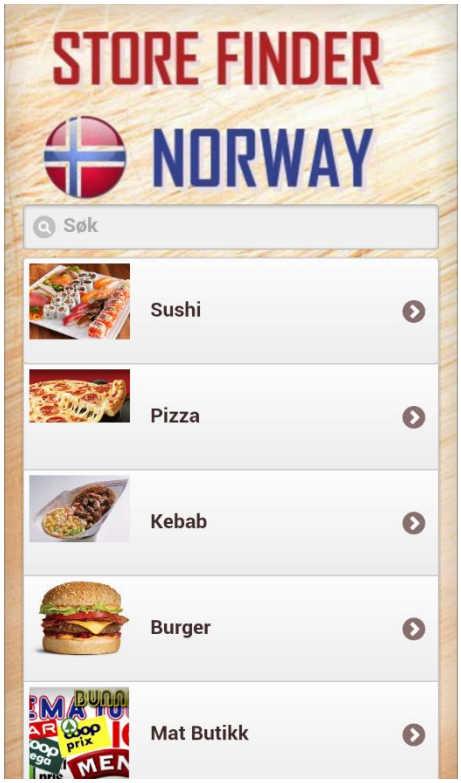
Eksempel på hvordan vinduet ser ut etter å ha valgt reisemetode, sted å reise fra, og destinasjon.



Det er også mulig å bruke street view her ved å dra det gule person-ikonet over zoom-knappene nederst til høyre inn på en gate.

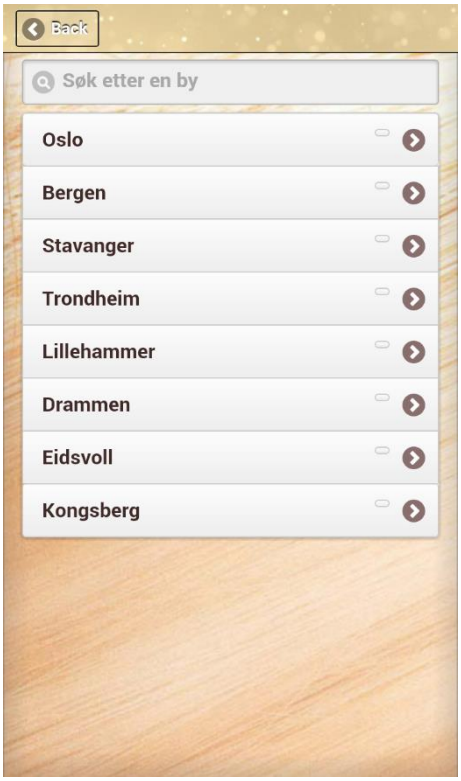
«HVA VIL DU SPISE»

Når den tredje knappen på forsiden trykkes, vises en liste over ulike typer mat. Her kan du finne informasjon om spisesteder når du ikke har noe konkret å søke etter. Denne funksjonen er ikke avhengig av en internett-tilkobling.



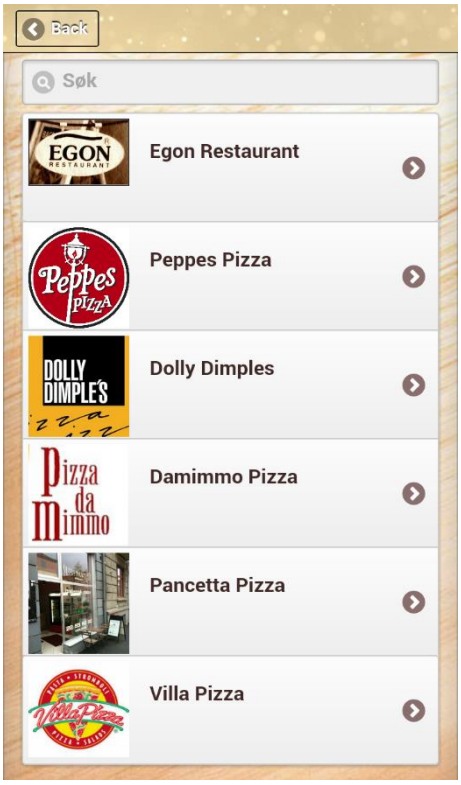
Du kan også søke etter matbutikker og kjøpesentre.

Etter at du har valgt et av alternativene, hoppes det videre til en liste over byer. Velg hvilken by du er i for å vise relevante resultater.



NB: Kun Oslo er lagt inn for øyeblikket.

Etter å ha valgt en by, vises det en liste over aktuelle spisesteder i byen. Trykk på et av spisestedene for å se informasjon.



Her får du sett kontaktinformasjon, adresse, og åpningstider. Hvis bedriften har en hjemmeside kan du finne linken til den her, samt en knapp som finner bedriften på Google Maps.

INSTALLASJON AV CORDOVA OG ANDROID EMULATOR I NETBEANS 8.1 (WINDOWS)

Her følger en veiledning til hvordan man installerer de nødvendige programmene og komponentene for å få samme oppsett som gruppen hadde i utviklingen av prosjektet.

Dette er beregnet for arbeidsgiver for vedlikehold og videreutvikling av løsningen.

1. Installer NetBeans 8.1
2. Last ned node.js v4.4.4 → https://nodejs.org/en/
3. Gå til command prompt (cmd)
4. Tast inn node --version
    - Er du bak proxy må du konfigurere den til port 8080
    - cmd: `npm config set proxy http://proxy:8080`
    - cmd: `npm config set https-proxy http://proxy:8080`
5. Sjekk om du har riktig konfigurasjon

    ```
    npm config list
    ```
6. Installer Apache Cordova

    ```
    npm install –g cordova
    ```
7. Oppdater Cordova

    ```
    npm update –g cordova
    ```
8. Sjekk versjon til Cordova

    ```
    cordova –version
    ```

9. Installer GIT

- https://git-scm.com/

10. Sjekk installasjon av git versjon:

- `git –version`

11. Legg GIT Path til Enviroment Variables

- Gå til Control Panel → System & Security → System → Change Settings. Da vil System Properties Windows dukke opp. Velg Advanced og klikk på Environment Variables.
- Klikk på User Variables, velg en Path og klikk på Edit knappen. Legges inn slik:

  `C:\Program Files\Git\bin\git.exe;C:\Program Files\Git\cmd`

12. Installer Java Development Kit (JDK):

- **www.oracle.com/technetwork/java/javase/downloads/**.
- Legg inn environment variables:

  C:\Program Files\Java\jdk1.8.0_11\bin
  C:\Program Files\Java\jdk1.8.0_11
- På terminal:
  `javac –version`

Installasjon av Android SDK Tools for å kjøre emulator

1. Installer Android bundle:

   http://developer.android.com/sdk/index.html#downloads

2. Legg Android path til environment variables

3. Skal se sånn ut:

```
C:\Development\adt-bundle\sdk\platform-tools;C:\Development\adt-bundle\sdk\tools
```

4. Sjekk om Android bundle er installert riktig

```
Adb --version
```

5. Åpne Android Studio. Gå til Android SDK Manager og installer nødvendige pakker:
   - Android (API 17)
   - Android (API 18)
   - Android (API 19)
   - Android (API 20)
   - Ekstra filer som støtter ulike plattformer du vil kjøre til Cordova.

6. Sjekk om alle PATH på Environment Variables er riktig lagt inn:

```
Echo %PATH%
Echo %JAVA_HOME%
```

Åpne NetBeans 8.1

1. New project
   - HTML5/JavaScript folder
   - HTML5/js Application
   - Navn på prosjektet ditt
   - No Template
   - Finish

2. Høyreklikk på prosjektnavn → Project Properties

3. Cordova

    - Create Cordova Resource
    - Plugins: legg til Geolocation, Globalization, Network Connection, Media, File API, Device API, Dialogs, File Transfer
    - Helt nederst er det en Mobile Platform: sjekk om Android SDK location stemmer. Ved siden av så er det Node.js: legg til source og path til NPM.

4. Legg inn nødvendig bibliotek:

    jquery,jquery-mobile

5. Gå til Index.html

6. Legg inn HTML import fra css, jquery-mobile.

7. Legg inn Google Maps test

    https://developers.google.com/maps/documentation/javascript/tutorial#Audience

8. Prøv å kjøre den.

9. Sett inn din egen API key. Hvordan denne lages er forklart nedenfor.

Opprett Google API key

1. https://developers.google.com/maps/documentation/javascript/get-api-key

2. Create new project

3. Lag browser key

4. Kopier key du har fått til HTML'en din.

5. Skal se sånn ut:

```
script async defer

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCkUOdZ5y7hMm0yrcCQoCvLwzdM6M8s5qk&callback=initMap">

</script>
```

Åpne command prompt (cmd). Må være administrator.

1. Gå til mappen din

   ```
   cd c:\users\...
   ```

2. Installer plattformer

   ```
   cordova platform add ios
   cordova platform add android
   cordova platform add wp8
   ```

3. Cordova build

4. `cordova emulate wp8`

   eller f.eks. `android` i stedet for `wp8` for å kjøre emulator.

5. For å teste til device:

   `cordova device run wp8`

   eller

   `cordova device run android`

For mer informasjon om hvordan dette skal fungere, eller hvis det oppstår problemer med punktene ovenfor:

https://evothings.com/doc/build/cordova-install-windows.html

https://netbeans.org/kb/docs/webclient/cordova-gettingstarted.html

https://cordova.apache.org/docs/en/4.0.0/guide/overview/index.html

https://netbeans.org/kb/docs/webclient/cordova-gettingstarted.html

Del 7 – Appendiks og vedlegg

BIBLIOTEK

HTML STRUKTUR	FORKLARING
<!DOCTYPE>	Erklæring på hva som kjøres til browser
<Html>	Brukes til å definere at dette er et HTML-dokument
<Head>	Brukes i HTML for metainformasjon. Tittel og importering av ulike biblioteker inn
<Body>	Hvordan siden skal se ut og kjøres. Inneholder f.eks. tekst, link, bilder

HEAD	FORKLARING
Title	Bestemmer hva tittelen i nettleseren skal hete
Meta	Beskriver metadata om HTML-dokumenter
Meta charset	Spesifiserer tegnkoder for HTML, eks: utf8
Meta name	Spesifiserer navn på metadata
Link rel	Definerer linken, hva slags type det er
Link href	Definerer hvor filen skal hentes i mappe for CSS
Script src	Definerer at dette er et script som skal hentes fram
Script type	Definerer hva slags type fil dette er

BODY	FORKLARING
div	Definerer seksjon og brukes til element for CSS
id	Navn på objektet
ul	Lage en liste
li	Lage punkter og orden inni ul
a	Hyperlink, brukes til å linke med andre sider
a href	Hyperlink med hvor det skal hentes
strong	Gjør skrifter mørkere
Img src	Henter plassering til bilder
H1/H2/H3	Definerer størrelse på skrift

JQUERY-MOBILE	FORKLARING
data-role=page	Siden som skal vises i nettleser
data-role =head	Lager en toolbar på toppen av siden, som tittel eller søkemotor
data-role=content	Her skal innholdet ligge
data-role=listview	Lager en liste
Data-theme	Hvordan bakgrunnen skal se ut
Data- inset	True / false for at det skal klikkes på
Data-autodividers	Skildre alfabetisk
Data-filter	For å legge inn felt for true/false
Data-filter-placeholder	For å legge inn navn på søk
Data-ajax=false	Skrur av Ajax-kobling til Google Maps, så det ikke vises hvit skjerm
Data-transition=slidedown	Lag en listview slider

CASCADING STYLE SHEETS (CSS)	FORKLARING
Padding	Bestemmer tomrom mellom innhold og border top, right, left, bottom
Position	Bestemmer hvor det skal ligges, absolute, relative
Top	Ligger øverst, bestemmes med px
Right	Ligger til høyre, bestemmes med px
Bottom	Ligger til bunn, bestemmes med px
Left	Ligger til venstre, bestemmes med px
Important	Ignorer alle regler/syntaks. Tar det som det står
Background	Bakgrunn
Box-shadow	Skygge til boks
Background-color	Definerer farge til bakgrunn
Border	Bestemmer hvordan utkanten av en skal se ut
Border-radius	Størrelser på hvordan det skal se ut
Color	Farger
Background:url	Link til bakgrunn bilder
Width	Lengde på en object f.eks: map
Height	Høyde på en object f.esk: map
Text-indent	Bestemmer innrykk på tekster
Font-size	Bestemmer størrelser på tekst
Margin	Brukes til å generere tomrom rundt et element
Background posistion	Bestemmer bakgrunnens ståsted
.	Definerer class i CSS
#	Definerer en ID i CSS
Em	Brukes som pixel, 1em = 20 px

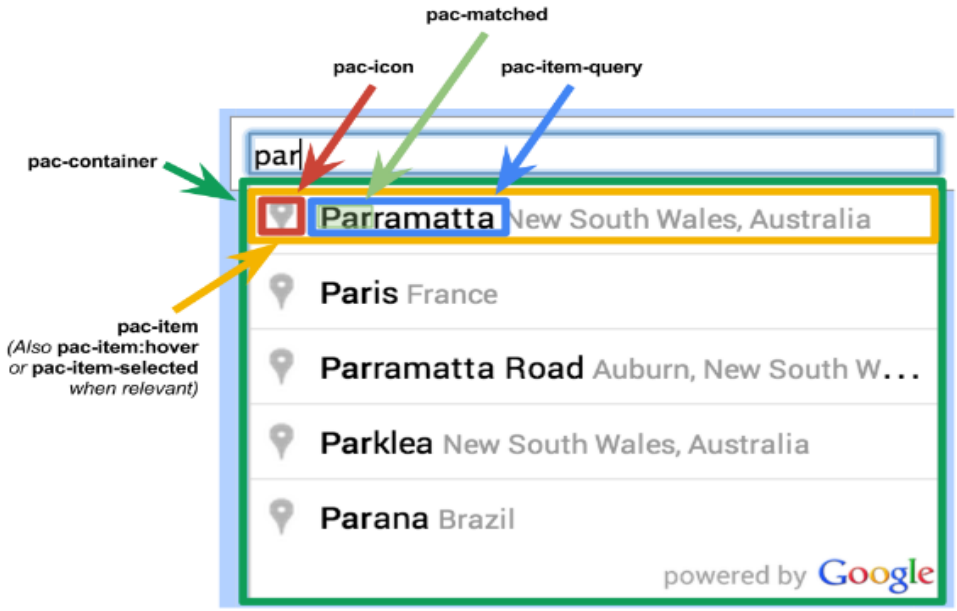
Pixel(px)	Størrelse på noe ting. 1 px= 0.021 cm

JQUERY-MOBILE CSS	FORKLARING
Ui	User interface
.ui-page	Brukes til lage dialog, lister osv. for hjemmesiden
.ui.body-c	Brukes til å bestemme tema av farge på innhold
.ui-corner-all	Bestemmer kant-radius til knapp/list visning
.ui-corner-tl	Bestemmer venstrekant radius
.ui-corner-tr	Bestemmer høyrekant radius
.ui-header	Bestemmer toppen av hjemmesiden som bilder og tekst
.ui-btn	Bestemmer knapper for <a>
.ui-listview	Bestemmer hvordan listevisning skal se ut
.ui-btn-active	For å vise hva som blir valgt
.ui-content	Hvordan innhold skal se ut
.ui-link-inherit	Arv fra linker
a.ui-link-inherit	Arv fra en annen link
.ui-block	Bestemmer hvordan rader og kolonner skal se ut
.ui-grid	Bestemmer kolonner og raders størrelser
.ui-bar-a	Bestemmer farger for bar i header, footer, osv.
.ui-icon	Lager ikon

JAVASCRIPT GOOGLE MAP	FORKLARING
InfoWindow	Viser innhold med tekst/bilder med melding inni kartet; det vises som regel inni Marker. Der vises det informasjon om lokasjoner.
New google.maps.	Definerer nytt objekt og beskrive hva den skal hente fra Google Service
SetPosition	Legg det til kartet
Latitude	Breddegraden
Longitude	Lengdegraden
GetCurrentPosition	Hent posisjonen
SetContent	Legg inn innhold/melding
GetCenter	Hent midtpunkt på kartet
Document.getElementById	Hente id fra html for å legge inn i objektet
SetBounds	Sette grensen
Places	Henter location informasjon fra Google biblioteket
Markers	Sette opp røde markører på kartet
SetMap	Gjengir lag på det angitte kartet, hvis det settes til null vil map ikke dukke opp
Anchor	Brukes som anker for Marker til infowindows(melding)
ScaledSize	Størrelse på hele bilder etter skalering, brukes til å strekke/krympe et bilde
Push	Legg element til tabellen og returnerer den nye tabellen
FitBounds	Sette viewport til beholder for den gitte grensen
Coords	Koordinater til en posisjon
TravelMode	Deklarerer/henter informasjon fra Google service for kjøreretning
MapTypeControl	Bestemmer hva slags type kart du vil bruke til false/true
Zoom	Bestemmer størrelse på kartet som vises (1-22)

Center	Sentral på kartet
Google.maps.DirectionsService	Kobler seg til Google Maps service og henter veiretning.
Google.maps.DirectionsRenderer	Gjengir retning hentet fra Google directions service
BindTo	Brukes i lytteren, funksjon til bind sammen med view til en model
AddEventListener	Legger den gitte lytteren funksjonen til den gitte hendelsens navn. Returnerer en identifikator for denne lytteren som kan brukes med google.maps.event…
Google.maps.TravelMode.WALKING	Deklarerer/henter informasjon fra Google service for gåretning
Place.geometry.viewport	Hvis stedet har en posisjon, legge til kartet
AddListener	Legger til en lytterfunksjon
Getplace	Returner detaljer om lokasjonsinformasjon, hvis det blir hentet fram riktig av koden
Navigator	Navigerer breddegraden og lengdegraden på kartet
getCenter	Returnerer posisjon til midten på kartet

Google Cascading Style Sheets



Bilde hentet fra Google (*ref: Google Developers*).

GOOGLE CSS CLASS	FORKLARING
Pac-container	Det visuelle elementet som inneholder en liste over steder og returnerer lokasjoner som den tror er riktig posisjon for søk
Pac-icon	Viser ikon
Pac-item	Bestemmer hvordan tegn inne i søkeboks skal se ut
Pac-item:hover	Vises kontrast farge når man har musa eller finger over den
Pac-item-selected	Vises kontrast farge når den blir valgt ved bruk av mus eller piltast på tastatur
Pac-item-query	vises alternativt som land/sted med et annet f.eks kursiv, fet skrift.
Pac-matched	Lyser opp ordet du søke i alternativ dropdown lister enten farger eller sterk skrift

ORDLISTE

A

Apache Cordova

Et rammeverk for utvikling av mobile applikasjoner.

API

Application Programming Interface. Et grensesnitt i en programvare som tillater at deler av programmet kan kjøres av et annet program.

B

Bug

En feil i et program.

C

CSS

Cascading Style Sheets. Definerer utseendet på filer skrevet i HTML (nettsider).

G

Geolokasjon

En prosess som finner ut hvor du befinner deg.

H

HiOA

Høgskolen i Oslo og Akershus.

HTML

Et markeringsspråk som brukes til å strukturere nettsider.

J

JavaScript

Et skriptspråk som tilfører dynamiske elementer til nettsider.

K

Kanban

En type smidig utviklingsmetode.

Klient

Et program som kjøres lokalt hos en bruker.

Kryssplattform

Programvare som kan kjøres i forskjellige operativsystemer.

P

Plugin

Et komponent som legger til ekstra funksjonalitet til et eksisterende program.

S

SCRUM

En type smidig utviklingsmetode.

Server

Tjener på norsk. Programvare som tilbyr tjenester til andre maskiner over et nettverk.

W

WebView

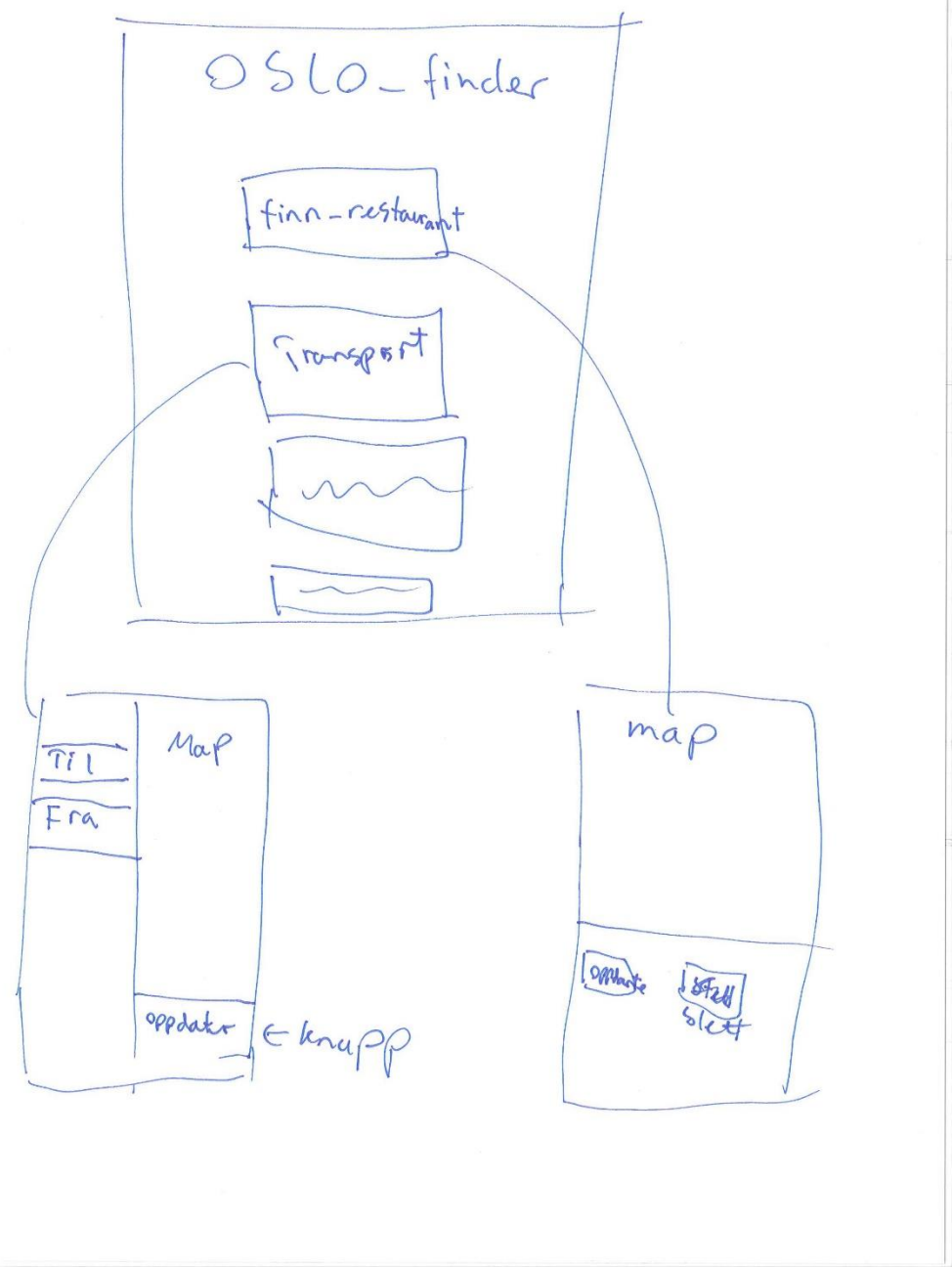
En nettleser inne i en mobil applikasjon.

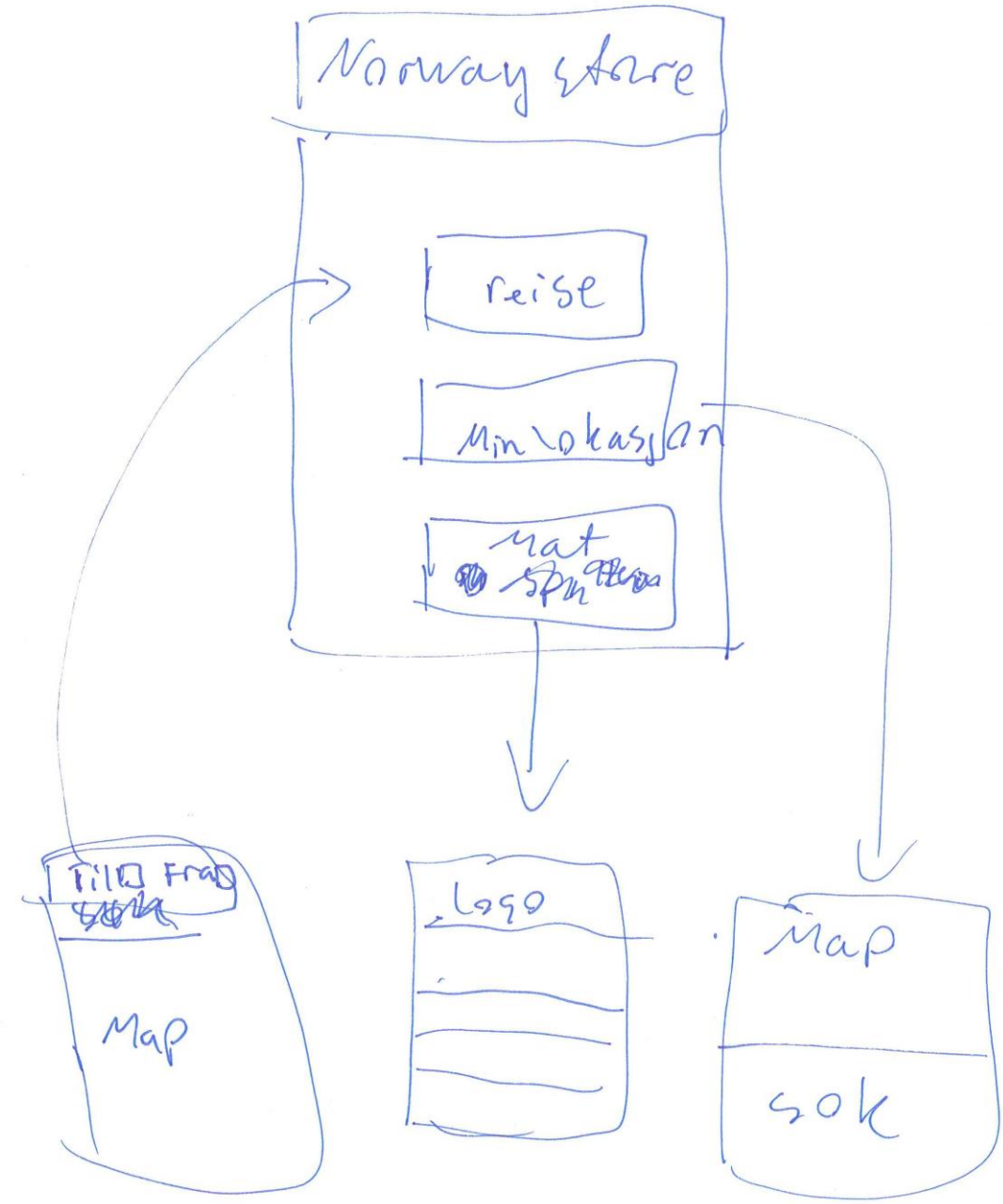
Wrapper

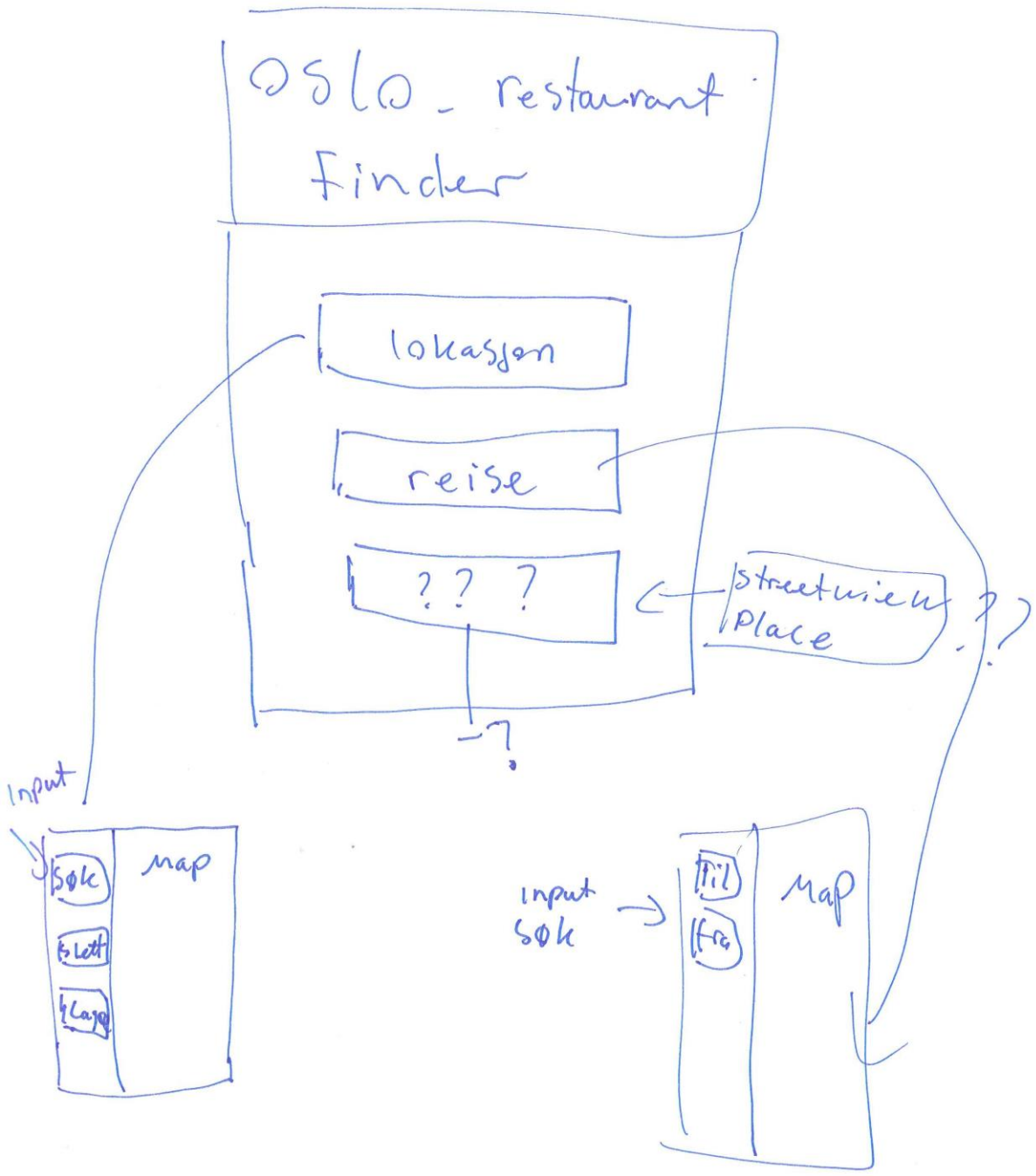
Data som pakker inn f.eks. et program slik at det kan kjøres i et annet program.

VEDLEGG

VEDLEGG 1 - SKISSER



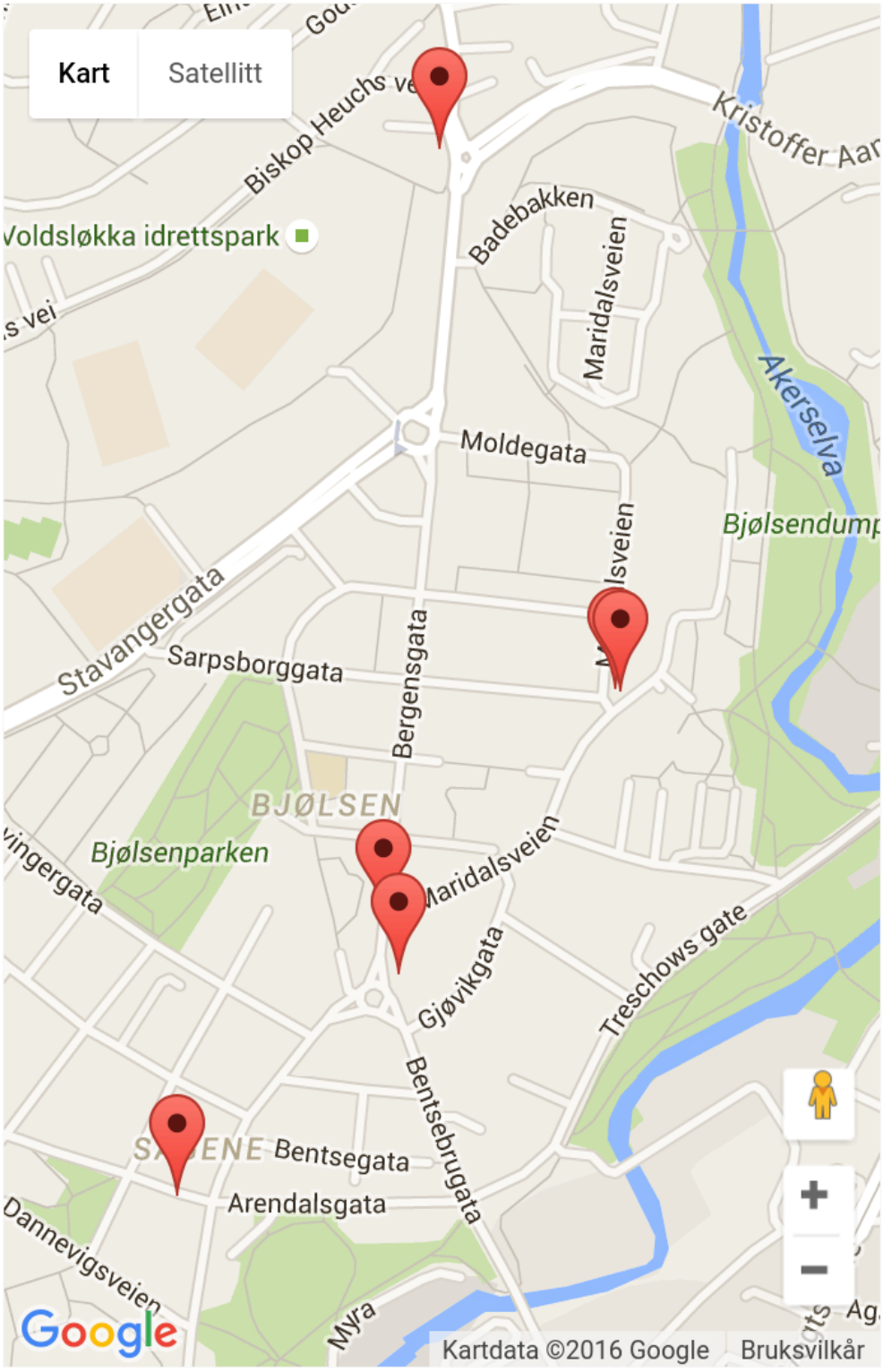

Norway store
reise
Min lokasjon
mat
Logo
Map
sok
Map


Oslo_ restaurant finder
lokasjon
reise
???
Streetview Place
input
søk
map
input søk
map

VEDLEGG 2 - GRENSESNITT


STORE FINDER
NORWAY
Hvor er jeg
Hvor vil du reise
Hva vil du spise

pizza


Kart
Satellitt
Biskop Heuchs vei
Kristoffer Aar
Badebakken
Maridalsveien
Akerselva
Moldegata
Bjølsendump
Stavangergata
Sarpsborggata
Bergensgata
BJØLSEN
Bjølsenparken
Maridalsveien
Gjøvikgata
Treschows gate
Bentsebrugata
Bentsegata
Arendalsgata
Dannevigsveien
Myra
Google
Kartdata ©2016 Google
Bruksvilkår

KILDEHENVISNING

- Kjempekjekt. Om smidig utvikling. Hentet 11.05.16 fra

  http://blog.kjempekjekt.com/2012/02/24/hva-er-smidig-utvikling/

- Machina. Om Kanban. Hentet 11.05.16 fra

  https://machina.no/smidig-utvikling-med-kanban/

- Google Developers. Illustrasjon av CSS. Hentet 19.05.16 fra

  https://developers.google.com/maps/documentation/javascript/images/place-autocomplete-css-diagram.png

- Cordova. Om arkitektur og plugins. Hentet 19.05.16 fra

  https://cordova.apache.org/docs/en/latest/guide/overview/

- Hvordan jquery-mobile fungere

  1. https://jquerymobile.com/
  2. http://www.w3schools.com/jquerymobile/
  3. https://learn.jquery.com/jquery-mobile/

- Google maps Javascript

  1. https://developers.google.com/maps/documentation/javascript/
  2. https://developers.google.com/maps/documentation/javascript/tutorial
  3. https://developers.google.com/maps/documentation/javascript/3.exp/reference
  4. https://developers.google.com/maps/documentation/javascript/examples/
  5. https://developers.google.com/maps/documentation/javascript/tutorials/

- **Google maps dokumentasjon av syntakser**

    1. https://developers.google.com/maps/documentation/javascript/places-autocomplete#places_searchbox
    2. https://developers.google.com/maps/documentation/javascript/places
    3. https://developers.google.com/maps/documentation/javascript/geometry
    4. https://developers.google.com/maps/documentation/streetview/

- Vi har også vært innom flere sider på stackoverflow og diverse utvikler sider for å lese om, men vi har ikke funnet en måte å finne det fram på.