# Afstemningsapps i App Lab på code.org

Forløb til b-niveau i informatik

Detaljeret forløb i afstemningsapps til informatik på b-niveau med links til tekster, videoer og opgaver til app lab på code.org samt brug af diagrams.net til at tegne diagrammer.

Fordele ved app lab:

- Udvikling i browser, så der ikke skal bruges tid på at installere
- Opgaver, hvor eleverne kan gå frem i eget tempo.
- Mulighed for at oprette hold, så man kan følge med i elevernes progression
- Fin tredeling, så eleverne kan arbejde med kode, grænseflade og data for sig.
- God debugger, der både udpeger fejl i kode og giver bud på mulige årsager.
- JavaScript med mulighed for at programmere i blokke og i tekst

**Introduktion (½ time)**

1) Introducer forløbet og giv elever links til et par eksempler på afstemningsapps, de kan afprøve.
2) Inddel eleverne i grupper og sæt dem til at vælge/trække en case.
3) Introducer vandfaldsmodellen for eleverne med fokus på, hvordan modellen er særligt velegnet ved en lav grad af innovation, hvor en eksisterende proces kan understøttes digitalt.
4) Gennemgå kort redskaber og arbejdsprocesser i de enkelte faser i vandfaldsmodellen.

**Foranalyse (2 timer)**

1) Bed klassen om, at komme med input til, hvad man kan undersøge, for at app'en matcher behovet.
2) Gennemgå interessentanalyse og sæt grupperne til at lave en interessentanalyse af deres egen case. Introducer eleverne til portfolio-redskab og bed dem uploade den færdige analyse.
3) Opsummer geografisk og demografisk målgruppeanalyse samt gallups kompas og bed eleverne fortage en analyse for de primære interessenter i deres case. Grupperne skal lægge analysen i port folio sammen med en refleksion over betydningen for udformningen af app'en.
4) Introducer til dokumentanalyse af eksisterende websider og apps samt evt. tegning af rige billeder og interview af domæneeksperter. Grupperne laver egne analyser og dokumenterer dem i port folio.

**Kravspecifikation (½ time)**

Sæt grupperne til at lave en kravsspecifikation, der opsummerer forundersøgelsen:

- Afsender samt primære og sekundære målgrupper
- System: Skal alle brugere logge på? Skal man kunne se og/eller rette i tidligere afstemninger? Skal man kunne se, hvad andre brugere har stemt? Hvem skal kunne oprette, redigere og nulstille afstemninger?
- Data, der er behov for at indsamle, samt krav til spørgsmål og svarmuligheder: antal svarmuligheder, behov for illustrationer, lav/normal/høj stil, lixtal…

**Design og arkitektur**

Introducer til modellering af systemer – sammenlign med arkitektens tegninger og diagrammer og sæt det op imod en mere evolutionær proces, hvor man bygger en masse modeller.

**Mock-up (½ time)**

1. Repeter kort guidelines: AIDA; Gestaltlove, KIS, valg af farver og billeder
2. Vis og diskuter forskellige eksempler på gode og dårlige/vildledende afstemninger.
3. Sæt grupperne til at lave mock-ups i eks. Power Point for forside, resultatside samt mindst 2 afstemningssider. De skal uploade dem i port folio.

**Navigationsdiagram (½ time)**

1. Gennemgå eksempler på navigationsstrukturer med fokus på, hvor de hver især er gode at anvende. Bed grupperne tegne et diagram på diagrams.net over navigationsstrukturen i deres app og uploade den.
2. Introducer designretningslinjer med fokus på god mapping - eksempelvis Jakob Nielsens 10 usability-heuristikker eller Donald Normans Designprincipper.
3. Sæt grupperne til at opdatere mock-ups med fokus på:
   - Systemets status: er det synligt for brugeren, hvor hun er i afstemningen?
   - Kontrol og frihed: kan brugeren gå tilbage og rette svar?
   - Konsistens: har "ens" knapper samme placering og udformning?

**Use-case-diagram (½ time)**

1. Vis video om use-case-diagrammer
2. Opsummer trelagsmodellen og tegn i fællesskab et use-case-diagram, hvor logiklag og datalag tegnes som adskilte systemer, så kald fra program til database også fremstår tydeligt.
3. Sæt elever til at tegner use-case-diagram for egen app på diagrams.net og uploade det i port folio.

**Rutediagram (½ time)**

1. Vis video om rutediagrammer og introducer kort tegning af rutediagrammer på diagrams.net
2. Bed eleverne om at lave og uploade rutediagrammer for hhv. afstemning og logon ud det setup, de har tegnet i use-case-diagrammet.

**E/R-diagram (1-2 timer)**

1. Vis video om E/R-diagrammer og giv et overblik over modellering af relationelle databaser med entitetsklasser, attributter, relationer og nøgler.
2. Introducer til problemer ved redundant data samt evt. normalformer og sæt eleverne til at arbejde med diagrammer forskellige systemer. –Eks. ordresystem (varer, ordrer, kunder, byer) og/eller gymnasie (elever, hold, klasser, lokaler).
3. Sæt grupperne til at lave diagrammer over deres database med minimum tre entitetsklasser: brugere, administratorer og afstemninger.
4. Opdel klassen i matrixgrupper, og bed dem præsenterer og give respons på diagrammerne.
5. Saml eleverne i grupper igen og bed dem opdatere og uploade E/R-diagrammet.

**Udvikling af brugergrænseflade (2 timer)**

1. Introducer kort App Lab og de programmeringsstrukturer, eleverne kender fra c-niveauet.
2. Sæt grupperne gå ind i designmode og opstille siderne i deres app inkl. tekst og knapper, men uden menuen til at navigere mellem siderne! - På afstemningsskærmen skal der være: en label til spørgsmål med id'et *afstemningSpg* og tre knapper/billeder med flg. id: *afstemningEnig, afstemningLigeglad, afstemningUenig*.
3. Udvikl i sammen trin for trin en dropdownmenu, så den kan skifte mellem 3 sider:
    - Indsæt dropdown menu på forside med apptitel øverst og sidenavne under
    - Skift til ny side, ved klik i dropdownmenu
    - Udvidelse: kopier menu ud på alle sider
    - Afprøv og debug: hvorfor er det kun menu på forside, der virker?
    - Omstrukturering: sideskift ud i funktion, som startes af alle menuer. Funktion tager en parameter "sideTitel", som hentes med getText("menuID").
    - Afprøv og debug: husker valgt side, når man kommer tilbage til eks. forsiden. – Nulstil valgt punkt, så snart der er skiftet til ny skærm. - setProperty("menuID", "index", 0);
4. Sæt grupperne til at tilpasse grænsefladen til det design og den opbygning, de har skitseret i deres mock-ups og navigationsdiagram.

**Introduktion til JSON (1 time)**

Lesson 8 - Creating JavaScript Objects 1-6

1. Se filmen introduction to objects sammen og gå igennem lektion 8 opg. 2-4 sammen: Vis hvordan man kan hente forskellige egenskaber ved event og introducer kort til kobling af talkode og symbol .-
2. 8.5 Start op sammen: opret to navne og vis hvordan man skriver "students" ud med console.log
3. 8.5-8.6 Elever færdiggør 8.5 og går videre med 8.6
4. Opsamling del 8.6: Udfordr eleverne: hvad fås ved udskrift af: students[1].name ; students[1].grade… hvad skal man skrive for at finde Max's fravær?

**Kontaktapp med JSON - lister vs. objekter (1 time)**

Lesson 8 - Creating JavaScript Objects 7-12

1. 8.8: Afprøv app: Hvilke elementer er der? id'er? Hvad kan den kode der er? Hvilket objekt og egenskaber skal der bruges? - Brug ringbind/arkivskab med poster til at forklare behovet for at have listen contacts til objekterne contact1, contact2 & contact3. – Vis evt. eksempel på app med klassens lærere.
2. 8.9: Elever indtaste indtaster selv kontakter og kobler dem til array'et contacts.
3. 8.10: Fællesudvikling – trin for trin:
    - Via console.log vises hele listen "contacts" samt "currentIndex".
    - Hvordan vises kun oplysninger for kontakt på currentIndex? >> students[currentIndex]"
    - Hvordan vises kun navn for kontakt på currentIndex? students[currentIndex].name"
    - Hvordan får vi det vist på skærm?
4. 8.11-8.12: Elever arbejder selv med, hvor alle informationer og billede skal vises
5. 8.13-8.14 lav 8.13 og start op på 8.14 sammen. - Opret objektet newContact og hent oplysninger ind - newContact.name = getText("nameInput"); - og kobl på listen: appendItem(contacts, newContact);
6. 8.14-8.16 Elever laver opgaver selv og derefter samles op:
    - Hvordan ved bruger, at kontakt er indsendt?
    - Hvordan gemmer vi ved aktuelt index.
    - Hvordan viser vi den sidst-indsendte kontakt?!
    - Ekstra: Hændelsen "change" til at opdatere ved hver ny ændring/indtastning + vise prewiew image

**Kontaktapp med Database (1-2 timer)**

Lesson 9 - Permanent Data Storage & Lesson 10 - Reading Records

1. 9.1: se de to videoer som introduktion/repetition af, hvordan databasen bruges. - Spring 9.2-9.10 over, hvis eleverne har lavet en simpel database i App Lab på c-niveauet.
2. 9.11: Eleverne laver kontaktapp med database (kort). I opsamlingen byder eleverne ind med løsninger. Sammenlign evt. løsning med en databaseapp de har lavet på c-niveauet. Afslut med at gå ind i databasen og vis, at det blot er JSON-kode ved at klikke på debug i DB'en. –
3. Spring 9.12-9.13 over, men bed eleverne indtaste 3 kontakter i deres app og tjekke, at de er gemt i DB.
4. 10.1-10.4: Gå gennem opgaverne sammen med fokus på synkronisering.
5. 10.7-10.12: Elever arbejder selv med at læse data via id samt med gennemgang af data med for-løkke.
6. Lav en opsamling med fokus på for-løkke samt synkronisering:
    - hvad hvis to brugere tilføjer kontakter næsten samtidig? Hvilket problem kan opstå?
    - hvordan minimeres risikoen for at to kald overskriver hinanden? Kan det undgås?

**Logonsystem til afstemningsapp (2 timer)**

1. Vis App Labs eget sample app på et logonsystem og udpeg i fællesskab elementer fra kontaktappen, der kan bruges i et logonsystem
2. Skriv sammen med klassen delmål i en trinvis udvikling af logonsystem op på tavlen – eks:
    - Oprette en tabel - brugere - med brugernavne og password
    - Vis al data fra brugernavn, der matcher det indtastede brugerid.
    - Vis kun adgangskoden
    - Vis fejl, hvis adgangskoden er forkert
    - Gem brugernavn i variabel, hvis brugernavn er rigtigt
    - Hjælp hvis bruger har glemt at indtaste navn eller kode
3. Skriv et gruppenumre ud for første delmål og bed grupperne flytte bogstavet, når det er løst. – Sidder en gruppe fast eller er bagud, kan de få hjælp fra en gruppe, der er nået længere.

**Kontaktapp med Database – Opdatering af poster i database (1 time)**

Lesson 11 - Deleting Records + Lesson 12 - Updating Records

1. 11.1-11.2 Gennemgås kort sammen – kobl det til nøglerne i E/R-diagrammer / relationelle databaser.
2. 11.3-11.7 Eleverne introduceres til at slette kontakter. Gennemgå 11.7 sammen.
3. 12.2 Lav opgaven sammen – evt. få hurtige elever, der har lavet opgaven til at stå for det. Påpeg, at det er data i listen, de arbejder på - og så uploader de listen til databasen bagefter!
4. 12.2-12.7 Eleverne får systemet til at kunne opdatere kontakter - hurtige kan gå i gang med del 13.
5. Start opsamling med opgave 12.3 og vis, hvordan først sang og så sekunder opdateres.
    - Gennemgå opdatering af kontakter
    - Vis, at data slettes, hvis man henter data med et kald via id og opdaterer!
    - Løs 12.7 i fællesskab og vis, hvordan koden fra "Addcontact" kan kopieres og tilpasse ved blot at få id kopieret over og ændre felterne teksten hentes fra!
6. Spring over lektion 13, men vis, at der kan importeres/exporteres data via csv

*Sørg for at I har lavet opgave 12 i lektion 5, så I har en redigeringsskærm, hvor I kan opdatere eller slette kontakter: https://studio.code.org/s/csppostap-2018/stage/12/puzzle/5*

**Afstemningssystem: vis og opdater afstemning (4 timer)**

1. Vis App Labs sample app med spørgeskema og udpeg i fællesskab elementer fra kontaktappen, der kan bruges i et afstemningssystem.
2. Skriv sammen med klassen delmål i en trinvis udvikling af et logonsystem op på tavlen – eks:
    - Opret tabel - *afstemninger* - med kolonnerne *navn, spg, enig* og *uenig.* Tilføj én eller flere afstemninger inkl. resultater til test.
    - Læs tabel og vis spørgsmål og resultat
        - Hent første afstemning og vis spg, enig og uenig via console.log
        - Vis spg, enig og uenig i labels
        - Afprøv om også kan hentes data for øvrige afstemninger.
    - Skift mellem forskellige afstemninger ved klik på skærm
        - Opret variabel til afstemningsdata samt til afstemningsnr
        - Øg afstemningsnr ved klik og vis næste afstemning
        - Ved sidste afstemning skal systemet gå til første afstemning.
    - Visning og valg af afstemning i dropdownmenu
        - Indsæt punkt i afstemnings-menu via kode (options i funktionen Setproperty)
        - Indsæt flere punkter via liste i afstemnings-menu
        - Hent navne på afstemninger fra db ind liste og sæt den ind i menuen.
        - Kig i design og find egenskaben, der styrer punktet der er valgt i menuen (index).
        - Udskriv punkt med console.log, når et punkt i menuen vælges.
        - Få punkt overført, så den tilsvarende afstemning vises.
        - Evt. udvidelse/omstrukturering: system finder og viser automatisk afstemninger uanset antal.
    - Opdatering af stemmer
        - Ved klik øges enig/uenig med én: afstemningsdata[afstemningsnr].enig++;
        - Vis opdatering og send til database (Brug Update og indsæt afstemningsdata[afstemningsnr]I i stedet for det tomme objekt (krøllede parenteser/lilla felt), der står der som standard.
        - Evt. udvidelse: kun brugere, der er logget på, må stemme.
        - Evt. omstrukturering: fælles funktion til at opdatere/gemme stemme: gemStemme(valgKnap)
    - Tilpas udseende og struktur til mock up.
3. Skriv gruppenumre ud for første delmål og sæt grupperne i gang. Grupper der er hurtigt færdige sendes ud som konsulent.

**Afstemningssystem: vis afstemningsresultater i diagrammer (2 timer)**

1. 14.1-14.7 Elever introduceres til visualisering af data - hurtige elever kan prøve at implementere det i egen app
2. Opsamling med fokus på, hvordan data loades i 14.5 og indstilling af options via json-objekt i 14.6
3. Lad eleverne afprøve If-b Diagram ud fra database eller lignede visualisering af data og diskuter fordele og ulemper ved de forskellige typer af diagrammer i App Lab.
4. Klik ind i Toolbox og Data og tjek API ud for drawChart og drawChartFromRecords. - Det er enklest at styre at bruge drawChart og lave en tabel med den data, man vil vise.
5. Udvikl i fællesskab et lille loop, der overfører data fra databasen til en tabel med objekter i det format som draChart kan bruge.
6. Vis hvordan de kan indstille udseendet på tabellen via et json-object, der typisk kaldes myOptions.
7. Sæt grupperne til selv at tilpasse og videreudvikle visualisering af deres data.

**It-sikkerhed - cases med sikkerhedsbrud og passwords (4 timer)**

1. Introducer en model for risikostyring som Cybersecurity Framework af National Institute of Standards and Technology med faserne: Predict, Prevent, Detect, Respond, Recover.
2. Opdel i matrixgrupper og giv hver gruppe en case på sikkerhedsbrud, de skal undersøge og præsentere en løsning for, hvordan det kan håndteres i en eller flere af faserne i modellen. – Brug gerne cases, der relaterer sig til brug af passwords:

- Keyloggers på biblioteker og andre offentligt tilgængelige computere.
- Afluring af dankortkode over skulderen på folk
- Datalæk, hvor hackere har fået fat i brugernavn og passwords i en database.
- Brugere gemmer password på seddel ved computer eller som billede på telefon
- Brug af svage/hyppige passwords
- Hack af passwordsmanegere

3. Introducer til analyse af trusler med risikomatrix og bed hver gruppe om at placere deres case ind, når de holder oplæg.
4. Lav en opsamling med fokus på go brug af passwords. - Lav et nyt sikkerhedsmatrix, hvor I placere elevernes tjenester ind: mail, uni-login, nem-id, sociale medier, spiltjenester… Pointér at tjenester med høj konsekvens bør have eget stærkt password, mens man godt kan genbruge passwords på tjenester med lav/ingen konsekvens.
5. Lav i fællesskab et risikomatrix, hvor generelle trusler for afstemningssystemer udpeges og vurderes
6. Gennemgå truslerne én for én startende med dem der har en høj konsekvens og en høj risiko. – Overvej i fællesskab, hvordan de kan undgås, opdages og håndteres.
7. Afslut med at grupperne implementere enkelte løsninger:
    - Brugere må kun kunne stemme én gang
    - Afluring af kode hen over skulderen: kode vises ikke ved indtastning, men erstattes af stjerner: ******
    - Kræv og tjek, at passwords har en vis længde.

## Udvidelser

Eksempler på aspekter forløbet kan udvides med eller elevene selv kan arbejde videre med:

**Interaktionsdesign**

- Design og layout i spørgeskemaer.
- Forsøg med at påvirke brugere til bestemte valg.
- Vægtning af resultater ved flere spørgsmål
- Interaktive test ved eksempelvis folketings- og kommunalvalg.

**Synkronisering**

- Undgå, at bruger stemmer flere gange i tidsrummet inden svaret fra databasen når frem.
- Sikring af at data opdateres, hvis app'en ligger uden at blive brugt.
- Synkronisering, så flere brugere kan stemme på samme tid.

**Admin side**

- Oprette, opdatere, nulstille, slette afstemninger
- Oprette, opdatere, nulstille, slette brugere
- Afstemninger til bestemte brugere
- Forskellige brugertyper med forskellige rettigheder: gæst, bruger, moderator, admin
- Sikkerhed: egen app til admin;  2-, 3,- N-lags arkitektur…

**Sikring af passwords i database**

- Vis videoen passwords & hash functions på youtubekanalen Simply Explained.
- Opsummer pointer omkring en-vejsfunktion og hvordan de kan konstrueres.
- Vis video om modulo på Khan Academy og sæt elever til at lave de tilhørende opgaver i modulo.
- Implementer en egen simpel hash-funktion i App Lab, der både bruges når password gemmes og nå login tjekkes – princippet i funktion er vigt – ikke styrke!
- Introducer til problem ved mange ens password og regnbuelister
- Vis videoen password hashing, salts, peppers på youtubekanalen Seytonic.
- Implementer i App Lab genereret med randomnumber og gemt i databasen
- Evt. kan de introducere peber i koden, der blot dannes ud fra tidspunkt eller dato