**Byttehandel**



Billede er fra <http://valuabledirt.files.wordpress.com/2012/02/buy-sell-dice.jpg?w=600>

Informationsteknologi – Lyngby Tekniks Gymnasium

Lærer: Lone Marianne Østerlund

Rapporten er skrevet af: Simon Ringive

Rapporten skal afleveres mandag d. 23/4 2012 (kl. 9:00)

# Resume

Deadline tirsdag

Normalerformer

Beskrivelse af tabeller

Er bytning lovligt? Skal det undersøges

Indhold

[Resume 2](#_Toc322796087)

[Projektbeskrivelse 5](#_Toc322796088)

[Problemformulering 5](#_Toc322796089)

[Forslag til løsning af problem 5](#_Toc322796090)

[Sams synsvinkel 5](#_Toc322796091)

[Max synsvinkel 6](#_Toc322796092)

[Projektafgrænsning 6](#_Toc322796093)

[Arbejdsopgaver 6](#_Toc322796094)

[Udviklingsværktøjer 6](#_Toc322796095)

[Hvad udføres ikke 7](#_Toc322796096)

[Tidsplan 7](#_Toc322796097)

[Lidt om server programmering. 8](#_Toc322796098)

[bytOnline 9](#_Toc322796099)

[Databasen 9](#_Toc322796100)

[E/R diagram 9](#_Toc322796101)

[Relationerne 9](#_Toc322796102)

[Normalformer 10](#_Toc322796103)

[Tabellerne 11](#_Toc322796104)

[Oprettelse af tabeller 12](#_Toc322796105)

[Layout af hjemmesiden – stylesheet.css 14](#_Toc322796106)

[Side-for-side specifikation 15](#_Toc322796107)

[Forsiden 15](#_Toc322796108)

[Sikkerhed cross scripting er mod javascript, bare skriv om sikkerhed mod min database og ikke generelt 16](#_Toc322796109)

[Videreudvikling 16](#_Toc322796110)

[Konklusion 16](#_Toc322796111)

[Bilag 16](#_Toc322796112)

[Ophavsret 16](#_Toc322796113)

[Kode 16](#_Toc322796114)

[Bilag 1 installer.php, config.php 16](#_Toc322796115)

[Bilag 2 stylesheet.css 16](#_Toc322796116)

# Projektbeskrivelse

Computerspil er kommet for at blive. Det er en stor interesse inden for en meget bred målgruppe og der findes smarte metoder i dag til at anskaffe sig de nyeste spil online. Dette kan f. eks være gennem en klient som Steam – Men hvad med de ældre spil? Vi føler os alle sammen nostalgiske en gang imellem og det kan faktisk være morsomt at prøve de ældre spil. Men hvordan skaffer man disse?

Mange unge ikke har råd til at købe de nyeste spil hele tiden. Ifølge min overbevisning og egen erfaring, så vokser penge ikke på træerne og SU dækker ikke altid ens fritidsinteresser. Jeg har hørt grunde fra mine klassekammerater om at de ønsker at spare penge fordi et nyt spil snart udkommer, hvis jeg forsøger at planlægge en biograftur så lyder det; ”skal lige se om jeg har penge til det” eller ” jeg kan ikke være med til at spise bagefter, men biografen ka’ jeg lige klare”. Det er ikke for at skyde skylden på spil kun, men hvordan kan man både gå i biografen og spise bagefter og samtidig anskaffe sig nye spil, når man ikke har pengene?

## Problemformulering

Det et problem at man ikke har mulighed for at bytte sine computerspil når man ikke gider spille dem mere.

## Forslag til løsning af problem

Kort og konkret vil jeg vil lave et website hvor man kan bytte sine computerspil online og det skal være muligt at skrive en lille anmeldelse af computerspillende for at tiltrække andre ”byttere”. Jeg kalder det bytOnline for nu.

Lad os kigge på 2 eksempler og hvorfor de kan løse ovennævnte problemer.

### Sams synsvinkel

Sam elsker computerspil, for ham er det mere end bare at spille spil – han samler på dem. Han er aktiv i mange spillefora hvor han bedømmer spil og skriver anmeldelser af dem. Han falder over hjemmesiden bytonline. Han opretter sig straks som bruger og har allerede oprettet nogle spil på listen som han ikke gider spille mere. Han forventer til gengæld at få nogle lidt ældre spil tilbage, som han mangler til sin samling. En person ved navn max kontakter ham og vil gerne bytte Assassin’s Creed for GTA 4, Sam accepterer og modtager et par dag senere GTA 4.

### Max synsvinkel

Max kan godt lide at spille computerspil ved siden af skolen, men de nye spil er dyre og han har ikke penge til både computerspil, fester og fitness. Der for bliver han nødt til at ønske sig dem på hans fødselsdag og til jul. Det kan resultere i at han ikke kan være med i samtalen om nogle af spillene med hans venner, og hvis han køber de nye spil, så har han ikke penge til at gå i byen og feste. Han finder frem til en løsning på nettet hvor siden bytonline tilbyder ham at bytte computerspil. Han opretter sig som bruger og finder Assassin’s Creed på en af byttelisterne. Han tilbyder at bytte hans GTA 4 og Sam, personen han kontakter, accepterer og et par dage senere modtager han det nye Assassin’s Creed.

Ideen til selve hjemmesidens bytningssystem er, at den ikke selv fortager bytningen. Den skaber bare kontakten mellem ”byttere” som har lagt de spil op de gerne vil bytte. Dvs. de skal indbyrdes aftale hvordan de vil bytte spil. Den skal derfor laves på dansk og appellere til danskere i første omgang.

## Projektafgrænsning

For at projektet ikke skal blive for omfangsrigt så vil jeg begrænse det til følgende.

### Arbejdsopgaver

1. Installering af database

2. Database hvor man kan:

1. Oprette sig som bruger
2. Oprette spil man ønsker at bytte inden for pris kategorier og skrive en kort anmeldelse
   1. Se og slette spil man selv har oprettet
3. Se liste over alle spil der er tilgængelige at bytte med, som man kan sortere i efter pris og kategori.

3. Website med bruger interface.

### Udviklingsværktøjer

Jeg vil udvikle hjemmesiden i XHTML, JavaScript, PHP og MySQl. Jeg bruger udviklingsserveren WAMP(Windows, Apache, MySQL and PHP) som giver mulighed for at lave og teste webapplikationer med apache-server, MySQL og PHP lokalt. Her bruger jeg phpMyAdmin til at administrere min database og til at designe selve brugerinterfacet bruger jeg CSS.

Til selve kodningen bruger jeg Notapad ++ som er et godt værktøj til at skrive kode med.

### Hvad udføres ikke

Jeg vil ikke lave brugerprofiler (ligesom i facebook hvor man skriver om sig selv og interesser) fordi det ikke er yderligere vigtigt for formålet med siden.

Selve bytnings systemet.

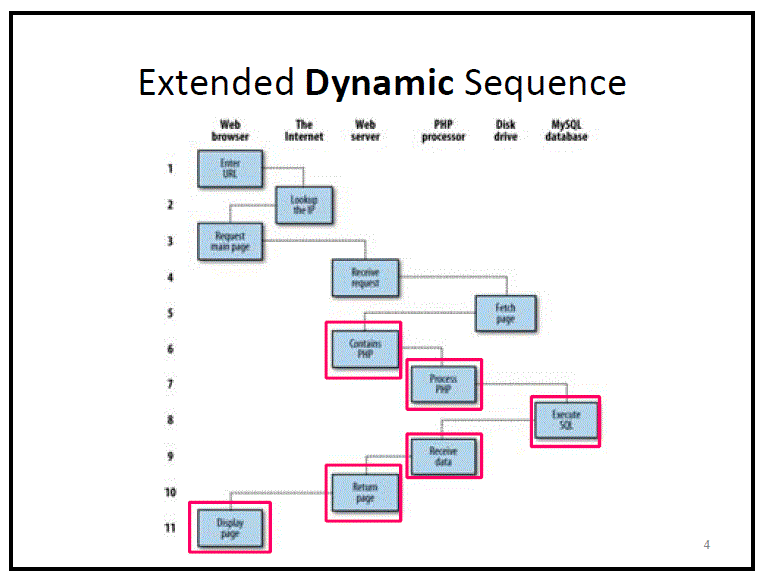
## Tidsplan

Jeg har lavet min tidsplan på uge basis, hvor arbejdsopgaven i enden af ugen er deadline. Der er så sat ekstra tid af til at arbejde med noget af det jeg ikke har nået i ferien.

|  |  |
| --- | --- |
| **Uge** | **Arbejdsopgave** |
| **8** | E/R diagram over database.  Normalformer. |
| **9** | Installering af database |
| **10** | Oprettelse af bruger og log ind |
| **11** | Oprettelse af spil og  Liste over spil der er tilgængelig at bytte med |
| **12** | Sortering plus slette spil funktion |
| **13** | Bruger interface med logo |
| **14 (Påskeferie)** | Finjustering + ekstra tid til ovennævnte ting  Rapportskrivning |
| **15** | Rapportskrivning |
| **16** | Rapportskrivning |
| **17** | Aflevering(mandag) |

# Lidt om server programmering.

En webbrowser (Internet Eksplorer,Google Chrome, etc.) som er en klient, kan fremvise HTML/XHTML filer, samt tilhørende objekter, som musik og billedfiler. XHTML siderne bliver afviklet direkte på klienten man nu har, men den kan ikke håndtere lagring eller udlevering af data fra f. eks en database. Den forstår ikke PHP filer. Til dette skal man bruge en webserver. En webserver kan nemlig håndtere lagring eller udlevering af data, hvis f. eks man skulle gemme login-informationer på en server. Dette gøres med PHP kode.



Figur : Eksempel på afvikling af en PHP side.

Figur 1 viser et eksempel på afvikling af en PHP side[[1]](#footnote-1). Brugeren indtaster f. eks en URL. anmodning om at gå ind på bytOnline sidens spil-liste (liste over spil brugere har oprettet). Webserveren modtager anmodningen og henter siden på det fil-lager(disk drive) den nu ligger i. Da siden indeholder en php-forespørgsel læser serveren dokumentet igennem og kører php-koden, for de beregningen/opgaver den har fået besked på. Det kan f. eks være at den skal hente de spil der er oprettet fra databasen (via funktionen ”sql\_query”). Når den har modtaget data, så returneres filen tilbage til klienten som HTML dokument. Klienten/webbrowseren ser kun resultatet af de beregninger webserveren modtager via html koden, og altså ikke PHP eller MySQL kode. Det kaldes derfor også for en dynamisk webside, da den laver disse beregninger og formulere det som HTML kode.

# bytOnline

Hjemmesiden i korte træk. (lille opsummering af hvad man kan på hjemmesiden)

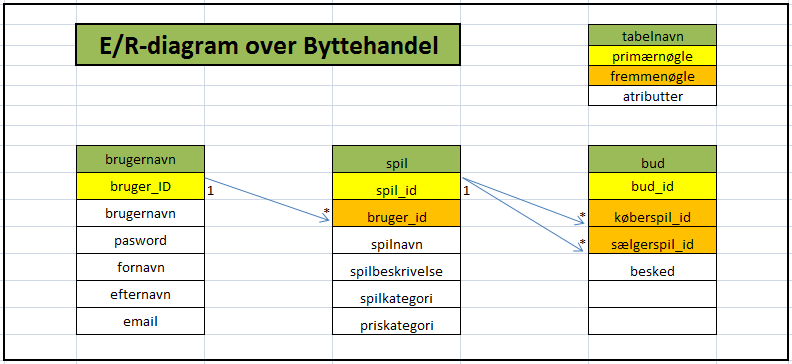
På bytOnline kan man oprette sig som bruger, logge ind og oprette spil man ønsker at bytte. Hvis ikke man ønsker at bytte et spil mere kan man slette det fra sin liste. Man kan også se og sortere i en liste over alle de spil der er oprettet.

Jeg vil i de følgende afsnit dokumentere min udvikling af hjemmesiden bytOnline.

# Databasen

Her kommer en forklaring af databasen for at give et overblik over hvad den skal indeholde.

## E/R diagram



Figur : E/R-diagram

## Relationerne

Som det ses på figur 1 så består databasen af 3 tabeller, en til brugeroplysninger, en til spil oplysninger og en til bud(altså håndtering af selve bytningen). Jeg vil her kort beskrive relationerne med eksempler fra Sam og Max, der bruger bytOnline.

Som bruger kan Sam oprette sit spil Assassin’s Creed, med et navn, en beskrivelse, en pris- og spilkategori. Han kan også oprette sit andet spil f.eks Rollercoaster Tycoon 3 på samme måde. Dvs. en bruger kan oprette mange spil, og et spil kan have en bruger – Dette er en 1 til mange relation.

Som bruger kan man både være sælgeren eller køberen af et spil når der skal oprettes et bud. Køberen betaler bare ikke med penge, men med et andet spil. Begge parter skal så acceptere, for at handlen kan gennemføres. F. eks så kan Max oprette et bud efter Sam’s Assassin’s Creed for hans eget GTA 4. Assassin’s Creed bliver her sælgerspil\_ID fordi Sam sælger spillet og GTA 4 bliver køberspil\_ID fordi max gerne vil købe Assassin’s Creed for GTA 4.

En bruger kan angive han vil bytte sit spil-A for en andens spil-B, en tredje bruger kan også angive at han vil bytte sit spil-C for samme spil-B – her er der en 1 til mange relation ved sælgerspil\_ID fordi spil-B godt kan være en del af flere bud. Det samme gælder omvendt at f. eks Spil-A kan være køberspil\_ID for både spil-B og spil-C for brugeren har oprette bud efter begge spil – Altså også en 1 til mange relation. Her skal det nævnes at samme bud aldrig kan have 2 sælger- eller køber\_ID.

Bemærk at jeg erstatter æ og ø med hhv. ae og oe i min kode.

## Normalformer

Når man laver en database vil man gerne undgå at data bliver gentaget eller at der opstår inkonsistens[[2]](#footnote-2). Det kan nemlig hurtigt komme til at fylde lagerplads, hvis der er mange brugere af et system. Jeg har derfor sikret mig at min database er på de 3 normalformer[[3]](#footnote-3)(forkortet 1NF, 2NF, 3NF).

**1. normalform** kræver at hver tabel i databasen har en primærnøgle, ellers ville man ikke kunne slå op i nogle af tabellerne uden. Det andet krav er at tabellerne kun har en-værdi-attributter. Flerværdi-attributter kan skabe problemer hvis man f. eks skal hente, indsætte, slette eller ændre dele af en flerværdi-attribut.   
 Min database overholder begge krav og er derfor på 1NF. Det kan diskuteres om f. eks et spil kan have flere kategorier som derfor udgør en flerværdi-attribut, men det har jeg valgt at det ikke skal kunne. Der er også tale om at attributter ikke må have NULL værdier som kaldes stærk normalform[[4]](#footnote-4). Dette har jeg overholdt, fordi der ikke er nogen steder i min database, hvor jeg syntes at der burde være mulighed for NULL værdier. Så min database er faktisk på stærk 1NF

**2. normalform(skal være på 1NF)** kræver at der ikke er nogen sammensatte kandidatnøgler(nøgle bestående af to eller flere attributter) i databasen. Hvis der er en sammensat kandidatnøgle så skal alle andre attributter i den pågældende tabel afhænge af den sammensatte kandidatnøgle. (gælder for alle sammensatte kandidatnøgler). Ellers ville der kunne opstå problemer med inkonsistens, hvor en mulig løsning ville være at splitte tabellen op i stedet.  
 Dette overholder min tabel da der slet ikke er nogen sammensatte kandidatnøgler, og den er derfor på 2NF.

**3. normalform(skal være på 2NF)** kræver at ingen af tabellens attributter afhænger indirekte af en kandidatnøgle. Hvilket igen kan skabe inkonsistens.   
 Dette overholder min database også og den er på 3NF.

Dvs. min database er på de 3 normalformer og der burde derfor ikke opstå inkonsistens, eller anden form for pladsspil.

## Tabellerne

## 

Figur : Tabellerne

På figur 2 ses de 3 tabeller jeg har i min database uddybet med datatype(størrelse) og andre egenskaber. Den har selv indsat et svensk charset, men derfor understøttes æ,ø og å stadig, så der er ingen problemer.

Generelt alle steder der skal stå kortere tekster, bruger jeg varchar datatypen. Grunden til nogle at de tilladte størrelser for varchar er store, er fordi brugeren selv skal indtaste værdier. Jeg bruger derfor cross-scripting hvor jeg erstatter ”<” og ”>” tegnene med ”&lt; ” og ”&gt;” (Se afsnit om sikkerhed). Dette fylder 4 gange så meget, så der skal være plads til 4 gange så stor en mængde. F. eks hvis brugeren skriver ”<” tegnet 30 gange i fornavn attributten (eller 16 ”<” i brugernavnsfeltet el. 70 ”<” i spilnavn attributten). Nogle spilnavne er ret lange og har derfor en endnu større tilladt tekststreng. Da jeg hash-kryptere passwordet (se afsnit om sikkerhed) så de vil altid fylde samme længde.

Til spilbeskrivelse i spil-tabel og besked i bud-tabel, har jeg givet dem datatypen text, der kan være en ubegrænset længde[[5]](#footnote-5). Dvs. der er en begrænsning men den er meget stor og kommer ikke til at blive overskredet.

De forskellige ID, er af datatype INT (dvs. hele tal uden decimaler) med maksimal længde 10 så der er mulighed for mange brugere. De er sat til AUTO\_INCREMNET så de selv generer et nyt ID hver gang en bruger oprettes. Brugernavn og e-mail er også gjort unike, da der ikke må være 2 af de samme.

Bemærk at jeg har lavet en isAdmin som er en tinyint. Denne kan være 1 eller 0 og angiver om man er bruger(0) eller administrator(1). Jeg har ikke indført nogle administrator funktioner endnu, men den er god at have for fremtidig kodnings skyld.

## Oprettelse af tabeller – installer.php, config.php

(Se installer.php og config.php i bilag nr. 1)

Disse 2 filer opretter databasen. Man skal åbne installer.php først i sin browser, og indtaste brugernavn og password til MySQL. Config opretter nu hele databasen relationerne mellem tabellerne, samt en bruger som har fuld rettigheder til den, men kun den. Fordelen ved dette er at hvis nogen skulle få uønsket adgang til databasen, så kan det kun gå ud over den ene database.

Følgende kode bruger jeg til at oprette databasen.

|  |
| --- |
| // opretter database  **if**(!mysql\_select\_db("bytOnline")) {  **echo** 'Skaber database<br />';  mysql\_query("CREATE DATABASE `bytOnline` ;") **or** **die**(mysql\_error());  mysql\_select\_db("bytOnline") **or** **die**("Fejl!"); //databasen vælges  **echo** 'Database bytOnline oprettet<br />';  } **else**{  **echo** 'Database bytOnline eksisterere allerede<br />';  } |

Først tjekker jeg at databasen ikke allerede eksisterer. Derefter oprettes den med SQL koden CREATE DATABASE. Derefter opretter jeg forbindelse til den, fordi de forskellige tabeller skal laves herefter. Koden nedenfor bruger jeg til at lave brugertabellen.

|  |
| --- |
| $lavBrugere = " CREATE TABLE `bytOnline`.`brugere` (  `brugerID` INT( 10 ) NOT NULL AUTO\_INCREMENT ,  `brugernavn` VARCHAR( 64 ) NOT NULL ,  `password` VARCHAR( 50 ) NOT NULL ,  `fornavn` VARCHAR( 120 ) NOT NULL ,  `efternavn` VARCHAR( 120 ) NOT NULL ,  `email` VARCHAR( 120 ) NOT NULL ,  `isAdmin` TINYINT( 1 ) NOT NULL DEFAULT '0' ,  PRIMARY KEY ( `brugerID` ) ,  UNIQUE (  `brugernavn` ,  `email`  )  ) ENGINE = INNODB;";  mysql\_query($lavBrugere) **or** **die**(mysql\_error()); |

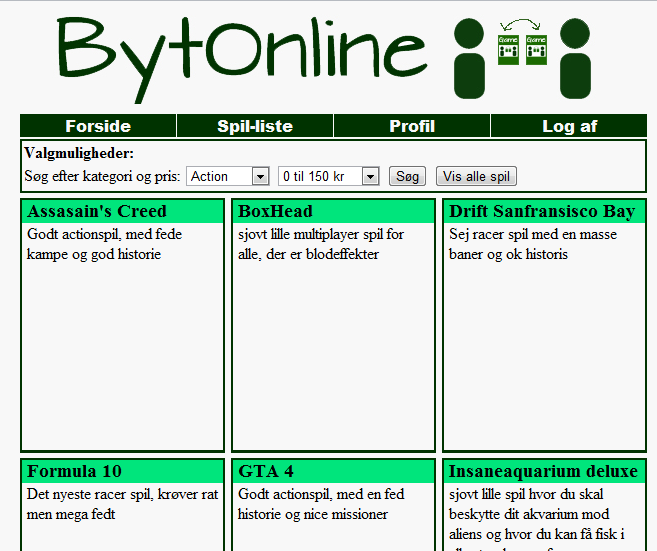
Her bruger jeg SQL koden CREATE TABLE til at oprette entiteten brugere, hvorefter jeg opretter attributterne med relevante datatyper. Til sidst angiver jeg primærnøgler og at brugernavn og e-mail skal være unikke. Koden nedenfor viser hvordan jeg laver relationen for brugerID i brugere- og spiltabellen.

|  |
| --- |
| $lavRelationForBruger = "ALTER TABLE `spil` ADD FOREIGN KEY ( `brugerID` ) REFERENCES `bytOnline`.`brugere` (  `brugerID`  ) ON DELETE CASCADE ON UPDATE CASCADE ;";  mysql\_query($brugereRelation) **or** **die**(mysql\_error());  mysql\_query($lavRelationForBruger) **or** **die**(mysql\_error()); |

Jeg ændrer i spiltabellen og angiver at brugerID er fremmednøgle. Derefter laver jeg en reference til brugerID i bruger tabellen og sætter den til at være ON DELETE CASCADE og ON UPDATE CASCADE. På den måde sørger jeg for at hvis der sker ændringer omkring BrugerID i brugertabellen, så sker de samme ændringer i spiltabellen.

# Layout af hjemmesiden – stylesheet.css

(Se stylesheet.css i bilag nr. 2)

Hjemmesiden er primært designet til dem som er interesseret i computerspil, hvilket er målgruppen jeg forsøger at nå. Derfor har jeg forsøgt at lave et design der passer denne målgruppe. 

Figur forside-designet Figur spil-liste-designet

De to billeder viser forsiden og spil-liste-siden. Jeg har gået efter et meget enkelt stil-design, med minimale farver, fordi en del af designet vil blive skabt af brugeren selv. Dette er jeg blevet inspireret af fra facebook.com og kongregate.com. Kigger man f. eks på facebooks.com på en profil, så giver billederne, brugeren har lagt op, en form for personligt design. Tager men dem fra, så er hjemmesiden meget enkel med den blå farve. Det samme gælder kongregate.com hvor de spil man kan spille, giver hjemmesiden et præg af forskellige farver, hvor den simple røde menu ligger tilbage i baggrunden. På samme måde havde jeg tænkt mig min hjemmeside skulle se ud. Forsiden giver en god ide om det, fordi billedet af spillet Uncharted3[[6]](#footnote-6) er med til at gøre hjemmesiden mere interessent. Den er ikke en direkte del af hjemmesidens design fordi det f. eks kunne være ugens spil, der ændrede sig efter det mest oprettede spil. På samme måde havde jeg tænkt mig at man på spil-liste-siden skulle kunne oprette et billede af spillet man vil bytte(se afsnit om videreudvikling). På den måde ville de firkantede bokse jeg har lavet, også være med til at skabe et varieret farverigt og dynamisk design. Designet bliver altså både mit og brugerens hvilket gerne skulle være med til at ramme målgruppen.

Logoet øverst har jeg selv designet, som skal forstille 2 personer der bytter spil. Jeg bruger herefter CSS kode til at designe resten af det grafiske på hjemmesiden. F. eks menu-bjælken.

Menu bjælken bliver designet ud fra følgende CSS kode i stylesheet.css nedenfor.

|  |  |  |
| --- | --- | --- |
| **Beskrivelse af .menu:**  .menu sørger for at sætte højde samt indrykning 4px til venstre på menuen. | **Beskrivelse af .menulink:**  .menulink sørger for at lave de 4 menu bokse.  Her sætter jeg teksttype, tekstfarve, boksens farve, og en masse indstillinger til størrelse og placering. | **Beskrivelse af .menulink: hover:**  .menulink:hover sørger for at gøre boksens farve lysere når musen er henover. |
| **.**menu **{**  **height: 25px;**  **margin-left: 4px;**  **color: #F8F8F8;**  **overflow: hidden;**  **}** | **.**menulink **{**  **color: #F8F8F8;**  **font-family:** 'Arial Black'**, Gadget, sans-serif;**  **display: block;**  **float: left;**  **width: 156px;**  **margin-left: 1px;**  **text-align: center;**  **background-color: #003300;**  **text-decoration: none;**  **}** | **.**menulink**:hover** **{**  **background-color:#006600;**  **}** |

Disse styles bliver kodet i klasse attributten i menu() funktionen i util.php (se bilag ??).

Til boksene på spil-liste-siden bruger jeg nogenlunde samme teknik som med menuen. Jeg bruger f. eks border-width, border-style, og border-color til at lave omridset og jeg bruger h3 tag til at lave overskriften hvor jeg har ændret lidt med CSS kode. Se evt. stylesheet.css i bilag 2.

# Side-for-side specifikation

bytOnline består af en del sider. Dem som brugeren kommer til at se er udstyret med logo og menuen i toppen, hvorefter sidens indhold vises. Brugeren vil altid kunne navigere rundt mellem disse og kommer aldrig ind på en side, hvor man ikke kan komme tilbage igen.

## Nyttige scripts

Disse filer, er ikke nogle brugeren ligger mærke til, men de har alligevel stor indflydelse på hjemmeside.

### util.php

(se bilag 3)

Uitil.php indeholder to nyttige funktioner Top(), bund() og menu() som automatisk bliver kaldt i top(). Top() og bund() skal inkluderes på alle siderne brugeren skal se. Top() sørger for at lave menuen og headeren samt starten på body. Alt synligt indhold skal placere herefter og skal afsluttes med Bund(), som afslutter tags.

### logindTjek.php

(se bilag 4)

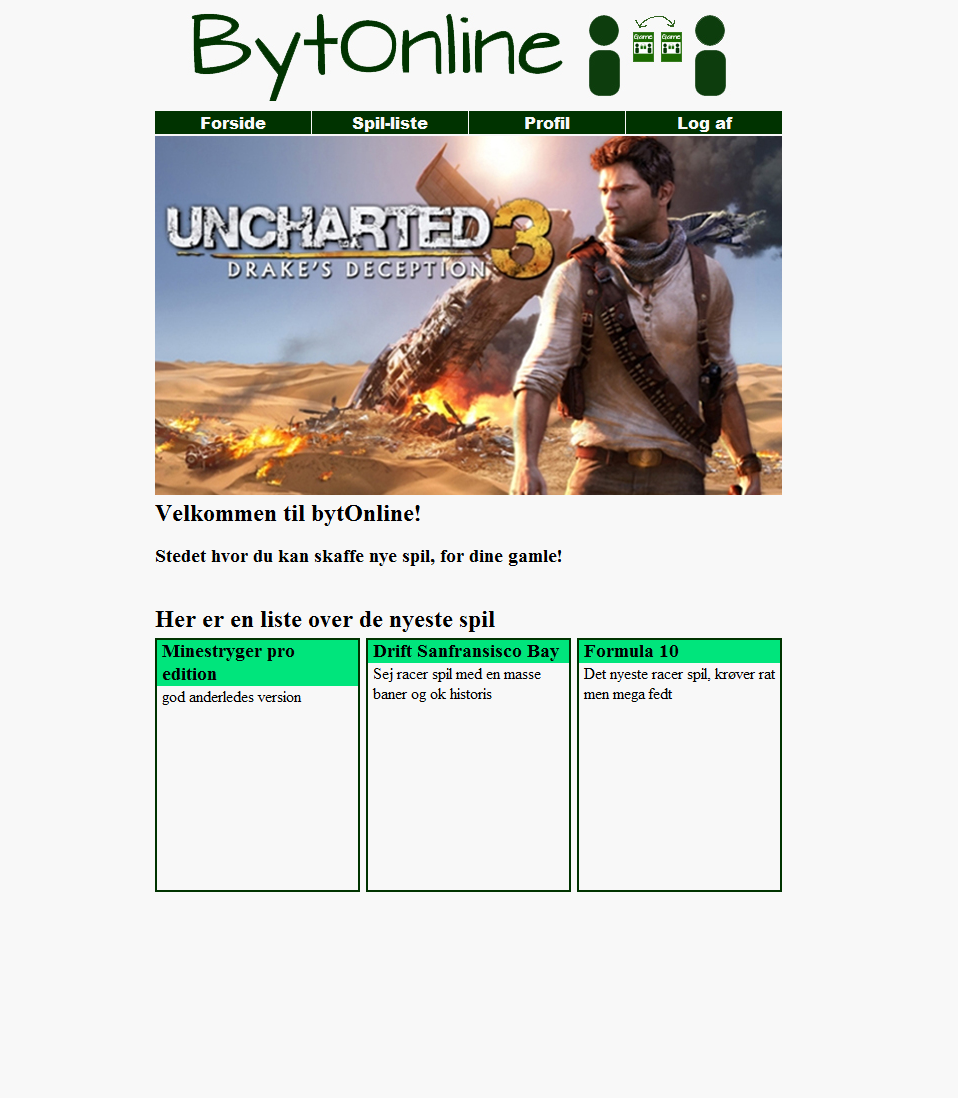
Denne skal inkluderes på sider der kræver man er logget ind, ellers bliver man henstillet til login.php hvor man efter at have logget ind bliver sendt tilbage til den oprindelige side ved hjælp af PHP\_SELF.

### connect.php

(se bilag 5)

Alle sider der skal hente, gemme, ændre eller slette filer i databasen skal have inkluderet denne her, som forbinder en til databasen. Dette gøres med den bruger der blev oprettet i config.php (se bilag 1) som kun har rettigheder til bytOnline databasen.

## Forsiden – index.php

(Index.php se bilag 3)  
Når en bruger går ind går ind på bytOnline, bliver forsiden vist først med en velkomst. Efterfulgt er der en liste over de 3 nyeste oprettede spil. Man kan altid komme tilbage til forsiden via menuen øverst.

Selve billedet og teksten er skrevet ved simpel HTML kode. Måden jeg viser de 3 nyeste spil på er ved et SQL udtræk fra databasen hvor jeg organisere de 3 nyeste oprettede spil i et array. Jeg kan derefter ved en simpel while-løkke tælle de første 3 pladser i arrayet op og jeg viser de nyste spil på samme måde som på Spil-listen.

Figur : Forsiden

## Spil-liste – spilListe.php

(se bilag 6)

Spil-listen viser alle spil der er blevet oprettet sorteret efter navn.

Brugeren kan vælge at sortere i listen efter kategori og pris i boksen lige under menuen ved at trykket på knappen søg. Han kan også trykke på knappen vis alle spil, som viser alle spil.

Figur : Spil-liste

Søgefunktionen fungerer via html attributten form action og input type="submit" som sender de valgte data til samme side (angivet i form action) til behandling af data.

Dette gøres ved følgende kode.(Se næste side)

|  |
| --- |
| <form action=**"spilListe.php?redirect=**<?php **echo** $\_GET['redirect']?>**"** method=**"post"** >  **Sg efter kategori og pris:**  <select name=**"spilKategori"**>  <option value=**"action"**>**Action**</option>  <option value=**"adventure"**>**Adventure**</option>  <option value=**"fps"**>**FPS**</option>  <option value=**"racing"**>**Racing**</option>  </select>  <select name=**"prisKategori"**>  <option value=**"green"**>**0 til 150 kr**</option>  <option value=**"yellow"**>**150 til 300 kr**</option>  <option value=**"red"**>**Over 300 kr**</option>  </select>  <input type=**"submit"** value=**"Søg"** />  <input type=**"button"** value=**"Vis alle spil"** onclick=**"location.href='spilListe.php?redirect=**<?php **echo** $\_GET['redirect']?>**'"**>  </form> |

Til indtastning af pris og kategori, har jeg lavet en drop-down liste, så brugeren skal ikke selv indtaste noget. Knappen søg, sender de valgte data til samme side, via GET-array’et, som så behandles. Fordelen er at en bruger så kan linke til en bestemt kategori og pris til en anden, i stedet for at den er skjult via POST-array’et. ”Vis alle spil”-knappen opdatere siden og alle spil bliver automatisk vist, hvis ingen værdier er sendt(via knappen søg). Visningen af spil foregår ved følgende kode.

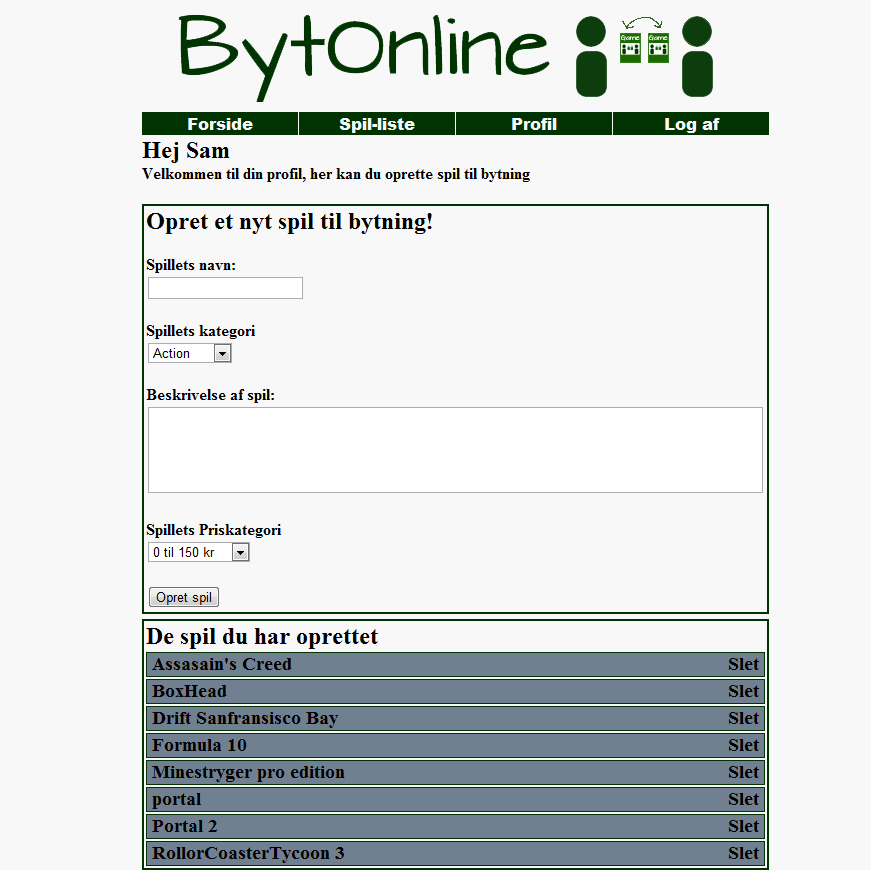
|  |
| --- |
| // hvis variabler er sat via knappen søg, så sorteres der i spil listen  **if** (**isset**($\_GET['spilKategori']) && $\_GET['prisKategori']) {  $spilKategori = mysql\_real\_escape\_string($\_GET['spilKategori']);  $prisKategori = mysql\_real\_escape\_string($\_GET['prisKategori']);    $resultat = mysql\_query("SELECT spilnavn,spilbeskrivelse FROM spil WHERE  spilkategori='".$spilKategori."' AND priskategori='".$prisKategori."'  ORDER BY spilnavn")**or** **die**(mysql\_error());  **while**($row=mysql\_fetch\_array($resultat)){  **echo** "<div class='spilboks'><h3><span>".$row["spilnavn"]."</span></h3>";  **echo** "<p>".$row["spilbeskrivelse"]."</p></div>";  } |

Hvis spilkategori og priskategori er sat, så bliver de escapet og gemt i samme variabelnavne via GET-array. Herefter bliver de relevante spil data fra databasen hentet ved at søge efter den valgte kategori og pris via SQL koden WHERE (se $resultat i if-sætning). Så bliver de sorteret efter navn og vist via PHP funktionen mysql\_fetch\_array() i en while-løkke. Her bliver CSS så brugt til at vise de hentede data som bokse. Det samme sker hvis ikke spilkategori og priskategori er sat, hvor alle spil vises i stedet.

## Profil – profil.php, opretSpil.php, sletSpil.php

(se bilag 7)

Profil siden er brugerens eget sted, hvor han kan oprette spil (han ønsker at bytte) ved at klikke på ”opret spil” knappen, han kan også se de spil han har oprettet eller slette spil han har oprettet ved at klikke på ”slet” teksten nederst ud fra det valgte spil. Hvis han ikke udfylder alle formene i oprtet spil, vises en fejlmeddelse øverst. Hvis ikke man er logget ind, sørger logindTjek.php for at sende en til login siden. Når brugeren er logget ind vises følgende.



Figur profil

Først finder jeg brugerens navn via brugerID’et som er gemt i $\_SESSION['LoggedIn']. På den måde kan jeg lavet en auto velkomst. Hvilket foregår ved SQL-koden SELECT fornavn FROM brugere WHERE brugerID = "'.$\_SESSION['LoggedIn'].'";

Oprettelsen af spil foregår nogenlunde på samme måde som søgning af spil på spil-liste siden. Det er samme ”form action” attribut og submit knap, og drop-down listerne er fuldstændig magen til, men brugeren skal også indtaste overskrift og beskrivelse her. Overskriften giver jeg en max længde 70 og angiver det er type=”text”, Til beskrivelse bruger jeg textarea hvor jeg sætter antal række og kolonner. Når opret spil knappen trykkes sendes værdierne gemt i variablerne til opretSpil.php siden. Hvis der er et sted man ikke har udfyldt vises en fejlmeddelelse øverst, som sendes via en error fra opretSpil.php. Se kode nedenfor.

|  |
| --- |
| // Tjekker at ingen felter er tomme  **if**(!$\_POST['spilNavn'] || !$\_POST['spilKategori'] || !$\_POST['prisKategori'] ||  !$\_POST['spilBeskrivelse']) {  header('Location: profil.php?error=1');  **exit**;  }  //Array som erstartter alle symbolerne '<' og '>' med '&lt;' og '&gt;'  $search = **array**('<', '>');  $replace = **array**('&lt;', '&gt;');    // Beskyttelse med Sanitize input  $spilNavn = mysql\_real\_escape\_string(str\_replace($search, $replace, $\_POST['spilNavn']));  $spilKategori = mysql\_real\_escape\_string($\_POST['spilKategori']);  $prisKategori = mysql\_real\_escape\_string($\_POST['prisKategori']);  $spilBeskrivelse = mysql\_real\_escape\_string(str\_replace($search, $replace,  $\_POST['spilBeskrivelse']));  $brugerID = mysql\_real\_escape\_string($\_POST['brugerID']);    $nytSpil = "INSERT INTO spil(spilnavn,spilbeskrivelse,spilkategori,priskategori,brugerID)  VALUES('".$spilNavn."','".$spilBeskrivelse."','".$spilKategori."','".$prisKategori."','".  $brugerID."');";  top();  // værdier bliver indsat via $nytSpil og service besked gives, samt tilbagelink  **if**(mysql\_query($nytSpil) **or** **die**(mysql\_error())){  **echo** "<div class='justering'><h2>Dit spil er nu oprettet.<br />  Du vil f堢esked hvis nogle nsker at bytte med dig.</h2></div>";  **echo** "<div class='justering'><a href='profil.php?redirect=".$\_GET['redirect']."'>  <h2>Tilbage</h2></a></div>";  } |

Først så tjekkes det at ingen felter er tomme, derefter erstattes ”<” og ”>” (se afsnit om sikkerhed) og der escapes mod sql-injection. Værdierne gemmes i variabler. Disse bliver så indsat i spil tabellen, med SQL-koden INSERT INTO spil hvorefter placering angives. Til sidst gives beskeden at spillet er oprettet og man kan gå tilbage til profilen.

Det allersidste brugeren kan inde på sin profil er at slette de spil han ikke ønsker at bytte længere. Spillene vises ved et h3 element med baggrunds farve, og en slet knap til højre. Visningen forgår på samme måde som med spilsiden med anden type CSS-kode. Trykker brugeren på slet, kommer en boks frem, som spørger om man er sikker. Så kommer man ikke til at slette ved et uheld. Dette gøres ved følgende JavaScript.

|  |
| --- |
| <!-- Funktion det beder om godkendelse, før den sletter en besked... -->  <script type=**"text/javascript"**>  <!--  ***function*** confirmation**(**id**)** **{**  ***var*** answer **=** confirm**(**"Sikker på vil slette dette spil?"**)**  ***if*** **(**answer**){**  window.location.href **=** "sletSpil.php?spilID="**+**id**;**  **}**  **}**  //-->  </script> |

JavaScript funktionen tager et id som parameter. Hvis man siger ja til at slette så bliver window.location.href sat til spilSpil.php. På sletSpil.php tjekker jeg om brugerens ID gemt i $\_SESSION['LoggedIn'] stemmer overens med det pågældende brugerID gemt i spil tabellen. Dette er for at undgå at en bruger kan slette andres spil.

|  |
| --- |
| $spilID = mysql\_real\_escape\_string($\_GET['spilID']);  //test om man er logget ind  $tjekBruger = "SELECT brugerID FROM spil WHERE spilID='".$spilID."';";  $tjekBrugerQuery = mysql\_query($tjekBruger) **or** **die**(mysql\_error());  $row = mysql\_fetch\_array($tjekBrugerQuery);  **if**($row['brugerID'] !== $\_SESSION['LoggedIn']){  header('Location: index.php');  **exit**();  } **else** {  //spil slettes og service besked samt tilbagelink gives.  top();  $sletSpil = "DELETE FROM spil WHERE spilID = ".$spilID.";";  mysql\_query($sletSpil) **or** **die**(mysql\_error());  **echo** "<div class='justering'><h2>Dit spil er slettet.<br /></h2></div>";  **echo** "<div class='justering'><a href='profil.php'><h2>Tilbage</h2></a></div>";  bund();  } |

Hvis brugeren må slette spillet bruger jeg SQL- koden DELETE FROM spil og så hvilket spilID der skal slettes. Herefter kan brugeren gå til bage til profilen hvis han vil. Ellers bliver man sendt tilbage til forsiden.

## Login – logind.php, logud.php



Figur : log ind form

Man kan komme ind på Logind.php siden via menuen eller hvis man går ind på en side der kræver man er logget ind. Klikker man log ind, tjekkes om brugernavn og adgangskode passer og hvis de gør, bliver man logget ind. Hvis man ikke har oprettet en bruger endnu kan man klikke på ”registrer dig her!” linket.

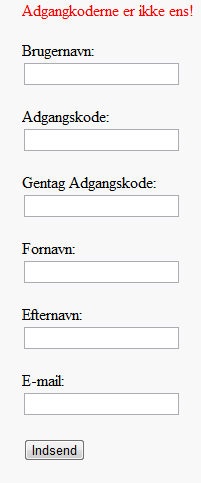
|  |
| --- |
| //Array som erstartter alle symbolerne '<' og '>' med '&lt;' og '&gt;'  $search = **array**('<', '>');  $replace = **array**('&lt;', '&gt;');    $tjekLogind = "SELECT \* FROM brugere WHERE brugernavn=  '".mysql\_real\_escape\_string(str\_replace(  $search, $replace, $\_POST['brugernavn']))."' AND password=  '".md5(mysql\_real\_escape\_string($\_POST['password']))."';";  $resultat = mysql\_query($tjekLogind) **or** **die**(mysql\_error());  **if**(mysql\_num\_rows($resultat) < 1) { // tjeker om brugernavn og password er rigtig  header('Location: logind.php?error=2&redirect='.$\_GET['redirect']);  **unset**($\_SESSION['LoggedIn']);  **exit**;  }  $row = mysql\_fetch\_array($resultat); //Henter valgte data fra databasen  $\_SESSION['LoggedIn'] = $row['brugerID']; |

Siden består af en simpel form attribut og Log ind knappen sender de indtastede værdier fra brugeren videre til loginQuery.php som kan ses i koden ovenfor. Her bliver der først tjekket for fejl, som at man ikke har skrevet noget, eller at brugernavn eller adgangskode er forkert ved mysql\_num\_rows funktionen. Hvis det er, så sendes man tilbage til logind.php med en relevant fejlbesked. Ellers bliver værdierne escapet og beskyttet mod JavaScript (se sikkerhed) hvorefter $\_SESSION['LoggedIn'] bliver sat til at være brugerens ID som jeg henter ved et SQL query.

Når man er logget ind kan man klikke log ud. Log ud sender en til logud.php, som sender en tilbage til siden man var inden på og destruere $\_SESSION['LoggedIn'] med session\_destroy();

# Registrerer – registrer.php, registrerQuery.php

Trykker man på registrer kommer man ind på registrer.php siden. På siden er der en masse felter man skal udfylde og en indsend knap. Dette er igen en form, med en submit knap der går til siden registrerQuery.php.



Figur registrer form

Her tjekker jeg for en masse fejl fra brugerens side, som f. eks om adgangskoderne er ens. Jeg bruger også samme fejlsystem til log ind og oprettelse af spil. Systemet er lavet på følgende måde.

|  |
| --- |
| **if** (**isset**($\_GET['error'])){  **switch**($\_GET['error']){    **case** 1:  **echo** "<span style='color:red;'>  Du mangler at udfylde nogle af formene.</span><br /><br />";  **break**;    **case** 2:  **echo** "<span style='color:red;'>  Adgangkoderne er ikke ens!</span><br /><br />";  **break**;    **case** 3:  **echo** "<span style='color:red;'>  Brugernavn eksisterer allerede.</span><br /><br />";  **break**;  **case** 4:  **echo** "<span style='color:red;'>  Den indtastede e-mail eksisterer allerede.</span><br /><br />";  **break**;    **default**:  **echo** "<span style='color:red;'>UKENDT FEJL</span><br />";  **break**;  }  } |

Jeg bruger en switch case til at tjekke hvilken fejlmeddelelse der er blevet sendt via GET-array’et og derefter vises den med rød skrift som også kan ses på billedet.

## Test af hjemmesiden

Jeg har løbet testet hjemmesiden under arbejdet. Dette er en god ide, for der er ikke nogen grund til at bygge videre på et skævt grundlag. Forstået på den måde at det bedst at tingen virker, før man går videre. Nogle kendte fejl jeg dog ikke har løst endnu er følgende.

### Kendte fejl

Disse fejl jeg poster her har jeg endnu ikke fået løst endnu.

Teksten der bliver vist i spilboksende, skal kun vise et bestemt antal bogstaver, hvorefter resten skal kunne læses når man klikker på den(er ikke lavet: se videreudvikling).

En anden kendt fejl, er at søgefunktions knapper altid viser det samme når man har trykket søg, i stedet for f. eks at vise det man søgte efter, på knappen. Dette er irriterende, da det indtil videre ikke står andet sted end øver hvad man har søgt efter

Hvis man kun skriver mellemrum i nogle forme så kan det se underligt ud.

Hvis ikke der er oprettet spil, vises tomme bokse på forsiden.

### Hyppige fejl

Jeg har haft flest fejl omkring syntakserne fordi det kan være svært at holde styr på, når man går fra HTML til PHP til MySQL på samme tid! Ellers har små fejl som at man har glemt at opdatere variabel navne ved ændringer og fejl når man skulle sende værdier med GET-arrayet.

### Bruger test

Jeg har fået nogen som ikke har hørt hvad min hjemmeside går ud på, til at teste min den. Dette er for at se om hjemmesiden umiddelbart giver mening, forstået på den måde at brugerne gør som jeg har tiltænkt de skal.

### Dumme testen

Jeg har med vilje forsøgt at gør alt det forkerte, for at se om jeg har kunnet smutte uden om log ind krævede sider, samt forsøgt i alle formene at oprette f. eks bruger uden fornavn osv. Undervejs kom jeg f. eks frem til at man som bruger, kunne slette andre brugers spil, hvilket jeg hurtigt fik løst.

Nogle af problemerne har jeg løst ved at inkludere logindTjek.php på nogle af siderne eller tjekket om login session er sat f. eks for at undgå at brugeren kan logge ind når, han er logget ind.

Det største problem jeg stødte på var når man skrev mellemrum i nogle forme, fordi noget af CSS koden ikke virker.

Ellers burde jeg have dækket de fleste sikkerhedshuller omkring brugerfejl.

# Sikkerhed

# Videreudvikling

Jeg synes selv at jeg er nået rimelig langt med hjemmesiden og endte med at have tid til at lave mere funktionalitet efter jeg var færdig med mine deadlines.

Det der skal fokuseres på ved videreudvikling er dog selve bytningssystemet, for det mangler for at hjemmesiden kan tages i brug. Ellers havde jeg tænkt mit at det skulle være muligt også at redigere i de spil, man har oprettet, hvis nu der var en fejl. Man skal også kunne oploade et billede af det spil man vil bytte (som jeg skriver om i layoutet) fordi det skal være med til at præge hjemmesidens design. Spilboksene kunne også gøres til links, så man kunne klikke på dem direkte, og spil-liste sidens søge funktioner kunne godt gøres mere avanceret.

# Konklusion

# Bilag

## Ophavsret

Stort set alt på hjemmesiden, med logo øverst har jeg selv fremstillet. Jeg bruger

## Kode

### Bilag 1 installer.php, config.php

### Bilag 2 stylesheet.css

1. Fra ”introduktion til serverprogrammering” dokument udleveret af læren. [↑](#footnote-ref-1)
2. Inkonsistens = oplysning der er lagret 2 steder i databasen som ikke stemmer overens [↑](#footnote-ref-2)
3. Ifølge Datalogi grundbogen af Martin Damhus. [↑](#footnote-ref-3)
4. Se fodnote 1 [↑](#footnote-ref-4)
5. Iføgle ”PHP og MySQL” bog af Kristian Langborg-Hansen [↑](#footnote-ref-5)
6. Billedet er fra <http://www.onlinemovieshut.com/wp-content/uploads/2010/12/uncharted-drakes-fortune.jpg> [↑](#footnote-ref-6)