**Byttehandel**



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

Informationsteknologi – Lyngby Tekniske Gymnasium

Lærer: Lone Marianne Østerlund

Rapporten er skrevet af: Simon Ringive

Rapporten skal afleveres mandag d. 23.4.2012 (kl. 9:00)

# Resumé

Dette projekt er mit eksamensprojekt i Informationsteknologi B,  
hvor jeg har valgt at arbejde med en dynamisk hjemmeside.

Hjemmesiden har jeg skrevet i XHTML, JavaScript, PHP og MySQL, hvor jeg bruger udviklingsserveren WAMP til at teste mine webapplikationer med apache-server.

Selve projektet går ud på at programmere en hjemmeside, hvor man kan bytte computerspil online, dog med visse afgrænsninger eftertidsrammen. Formålet er at gøre det muligt for en større målgruppe af brugere at bytte deres computerspil med hinanden.

Hjemmesiden har jeg designet brugervenligt med hensyn til layout og anvendelse, og fået det testet med positive tilbagemeldinger.

God fornøjelse!

Indhold

[Resumé 2](#_Toc322887664)

[Projektbeskrivelse 5](#_Toc322887665)

[Problemformulering 5](#_Toc322887666)

[Forslag til løsning af problem 5](#_Toc322887667)

[Sams synsvinkel 5](#_Toc322887668)

[Max synsvinkel 6](#_Toc322887669)

[Projektafgrænsning 6](#_Toc322887670)

[Arbejdsopgaver 6](#_Toc322887671)

[Udviklingsværktøjer 6](#_Toc322887672)

[Hvad udføres ikke 7](#_Toc322887673)

[Tidsplan 7](#_Toc322887674)

[Lidt om server programmering. 8](#_Toc322887675)

[BytOnline 9](#_Toc322887676)

[Databasen 9](#_Toc322887677)

[E/R diagram 9](#_Toc322887678)

[Relationerne 9](#_Toc322887679)

[Normalformer 10](#_Toc322887680)

[Tabellerne 11](#_Toc322887681)

[Oprettelse af tabeller – installer.php, config.php 12](#_Toc322887682)

[Layout af hjemmesiden – stylesheet.css 14](#_Toc322887683)

[Side-for-side specifikation 16](#_Toc322887684)

[Nyttige scripts 16](#_Toc322887685)

[util.php 16](#_Toc322887686)

[logindTjek.php 16](#_Toc322887687)

[connect.php 16](#_Toc322887688)

[Forsiden – index.php 17](#_Toc322887689)

[Spil-liste-siden – spilListe.php 17](#_Toc322887690)

[Profil – profil.php, opretSpil.php, sletSpil.php 19](#_Toc322887691)

[Login – logind.php, logud.php 22](#_Toc322887692)

[Registrerer – registrer.php, registrerQuery.php 24](#_Toc322887693)

[Test af hjemmesiden 26](#_Toc322887694)

[Kendte fejl/mangler 26](#_Toc322887695)

[Hyppige fejl 26](#_Toc322887696)

[Bruger testen 26](#_Toc322887697)

[Endelige sluttest 27](#_Toc322887698)

[Sikkerhed 27](#_Toc322887699)

[Videreudvikling 28](#_Toc322887700)

[Konklusion 28](#_Toc322887701)

[Bilag 28](#_Toc322887702)

[Ophavsret 28](#_Toc322887703)

[Kode 28](#_Toc322887704)

[Bilag 1 installer.php, config.php 28](#_Toc322887705)

[Bilag 2 stylesheet.css 28](#_Toc322887706)

# 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 computerspil man har liggende, som er gennemført? Dem man ikke gider spille længere, som andre måske gerne vil have fat i.

Mange unge har ikke råd til at købe de nyeste computerspil, der hele tiden udkommer på markedet. Ifølge min overbevisning og egen erfaring, så vokser penge ikke på træerne og SU’en dækker sjældent ens fritidsinteresser. Jeg har hørt fra flere af mine klassekammerater, at de ønsker at spare penge op, fordi et nyt spil snart udkommer. Hvis jeg f. eks 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 nye spil, men hvordan kan man både gå i biografen og spise bagefter og samtidig anskaffe sig computerspil, når man ikke har pengene?

## Problemformulering

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

## Forslag til løsning af problem

Kort og konkret vil jeg vil lave et website, hvor en større målgruppe kan bytte sine computerspil online. Det skal samtidig være muligt at skrive en lille anmeldelse af computerspillene, for at tiltrække andre ”byttere”. Jeg har valgt at kalde hjemmesiden for BytOnline.

Ideen til selve hjemmesidens bytningssystem er, at den ikke selv foretager 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 deres spil med hinanden. I første omgang skal den laves på dansk og appellere til danske computerspilinteresseret.

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 via posten.

### Max synsvinkel

Max kan godt lide at spille computerspil ved siden af skolen, men de nye spil er dyre. Han har ikke penge til både computerspil, fester og fitness. Derfor bliver han nødt til at ønske sig dem til sin fødselsdag og til jul. Det kan resultere i at han ikke kan være med i samtalen om nogle af spillene med hans venner. 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 dette. Et par dage senere modtager han det nye Assassin’s Creed.

## 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 priskategorier og spilkategori 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). WAMP 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 kode.

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.

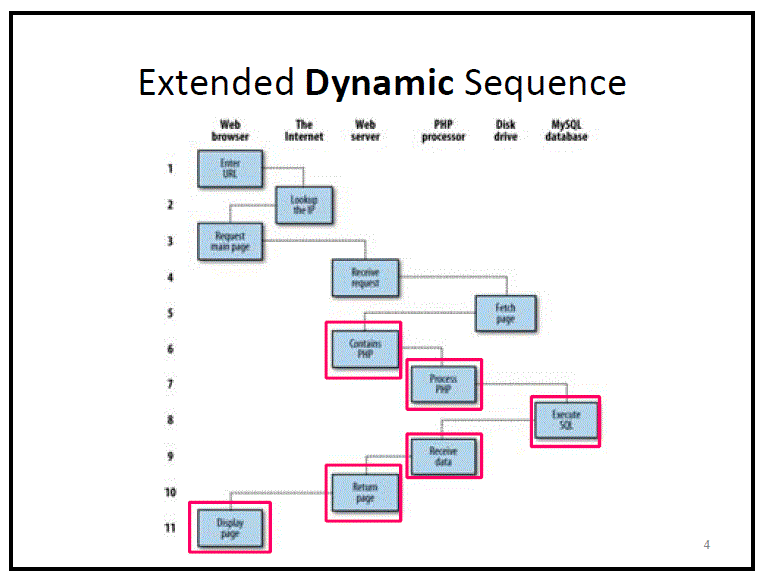
## Tidsplan

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, som jeg ikke har nået i ferien(påskeferien).

|  |  |
| --- | --- |
| **Uge** | **Arbejdsopgave** |
| **8** | E/R diagram over database.  Normalformer. |
| **9** | Installering af database. |
| **10** | Oprettelse af bruger og log ind system. |
| **11** | Oprettelse og slet af spil funktion. |
| **12** | Liste over spil der er tilgængelige til bytning samt sorteringssystem. |
| **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. HTML siderne bliver afviklet direkte på den klient man nu har. Den kan dog 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 beregninger/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 altså kun resultatet af webserverens beregninger via html kode, og derfor ikke PHP eller MySQL kode. Det kaldes derfor også for en dynamisk webside, da den laver disse beregninger og formulerer det som HTML kode.

# BytOnline

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

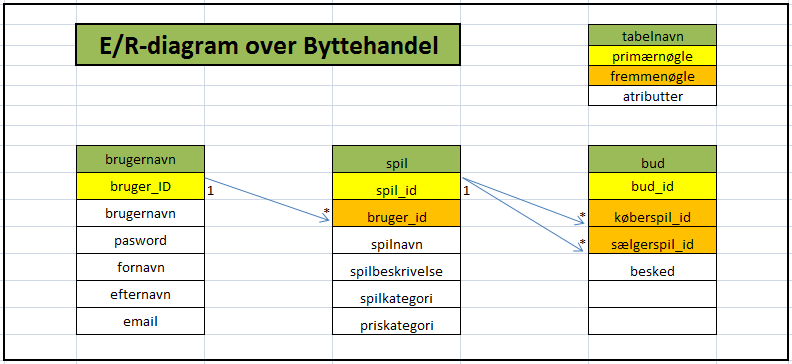
På BytOnline kan man oprette sig som bruger, logge ind og oprette spil man ønsker at bytte med. Hvis ikke man ønsker at bytte et spil mere kan man slette det fra sin liste. Man kan også se og sortere i en liste over alle de spil, der er oprettet.

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. Priskategorien er blot et estimeret bud på spillets værdi angivet af brugeren. 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 enten 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, at 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 fordi brugeren har oprettet et 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. 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 derfor 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.

Alle steder hvor der skal stå kortere tekster, bruger jeg varchar datatypen. Grunden til nogle at de tilladte størrelser for varchar er meget store skyldes, at brugeren selv skal indtaste værdier. Jeg beskytter mig derfor mod 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-krypterer adgangskoder (se afsnit om sikkerhed), vil de altid fylde samme længde og begrænsning betyder ikke så meget her.

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 unikke, 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 databasen. Config.php opretter nu hele databasen, relationerne mellem tabellerne samt en bruger, som har fuld rettighed 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.

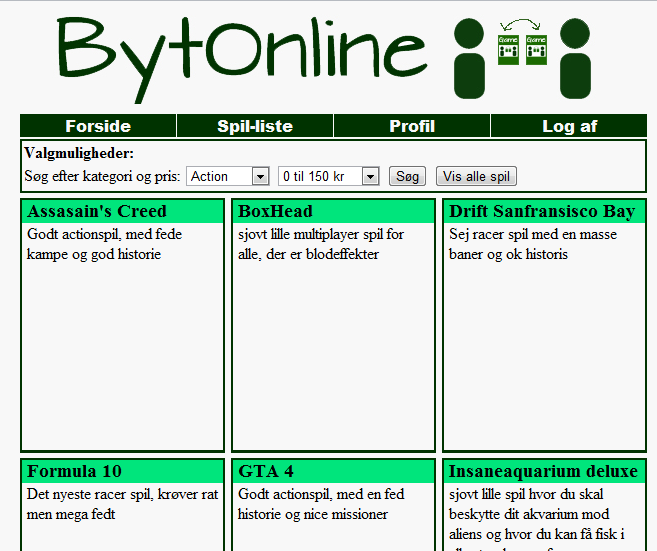
Koden nedenfor viser hvordan jeg laver relationen for brugerID i brugere- og spiltabellen.

|  |
| --- |
| $lavRelationForBruger = "ALTER TABLE `spil` ADD FOREIGN KEY ( `brugerID` ) REFERENCES `BytOnline`.`brugere` (  `brugerID`  ) ON DELETE CASCADE ON UPDATE CASCADE ;";  mysql\_query($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 i baggrunden.

På samme måde havde jeg tænkt mig, at 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 interessant. Den er ikke en direkte del af hjemmesidens design, fordi det f. eks kunne være ugens spil, som ændrede sig hver 7 dag. 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, som 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 menuen under logo.

Menuen 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 3), som er vist i PHP-koden nedenfor. Menuen med links og CSS, bliver kodet ved det der kaldes et associative array, som køres igennem en foreach, hvor det relevante CSS så bliver tilføjet.

|  |
| --- |
| $menuSider = **array**( 'Forside' => 'index.php',  'Spil-liste' => 'spilListe.php',  'Profil' => 'profil.php',  $loginT => $loginL.'?redirect='.$\_SERVER['PHP\_SELF']);// PHP\_SELF srger for at returnere den fil der lige er blevet krt  **echo** "<div class='menu'>";  **foreach** ($menuSider **as** $navn => $side) { // srger for at navn og links bliver vist og fungerer i menuen.  **echo** "<a class='menulink' href='".$side."'>".$navn."</a>";  }  **echo** "</div>"; |

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

# Side-for-side specifikation

BytOnline består af en del sider. De sider, som brugeren kommer til at se, er udstyret med logo og en menu 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. Men først vil jeg lige kort gennemgå nogle nyttige scripts.

## 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 tre nyttige funktioner Top(), bund() og menu(). Menu() bliver automatisk kaldt i Top(). Top() og bund() skal inkluderes på alle siderne brugeren skal kunne se. Top() sørger for at lave header og menuen samt starten på body. Alt synligt indhold skal placeres herefter og afsluttes med Bund(), som afslutter tags.

### logindTjek.php

(se bilag 4)

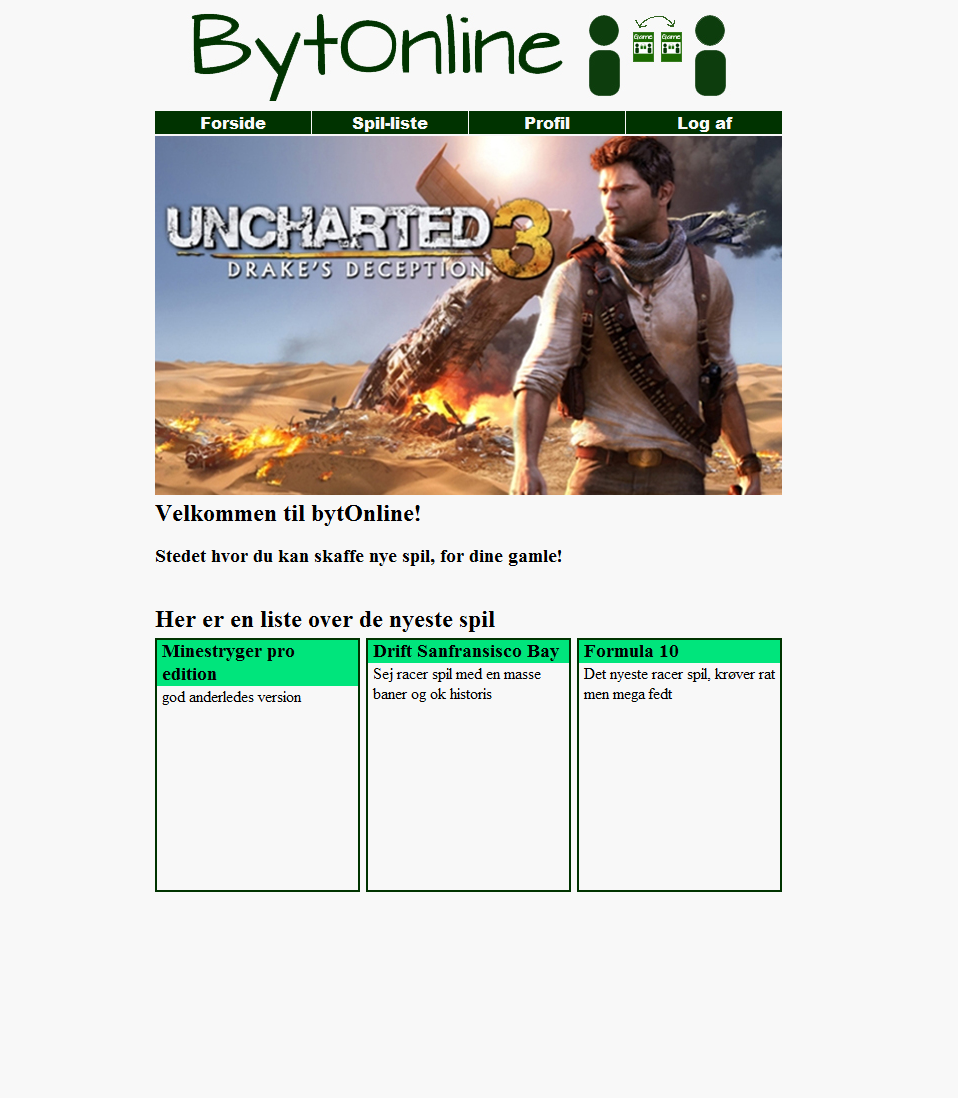
Denne skal inkluderes på sider, der kræver man er logget ind. Ellers bliver man henstillet til login.php, hvor man efter at have logget ind bliver sendt tilbage til den oprindelige side ved hjælp af en henstilling via PHP\_SELF. PHP\_SELF sørger for at returnere den fil der lige er blevet kørt. F. eks profil.php.

### connect.php

(se bilag 5)

Alle sider der skal hente, gemme, ændre eller slette filer i databasen skal have inkluderet denne her, som opretter en forbindelse 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 først vist med en velkomst. Efterfulgt er der en liste over de 3 nyeste oprettede spil. Man kan altid komme tilbage til forsiden via menuen øverst. Man kan kun komme videre til de 3 andre sider 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 finder de 3 højeste ID og organiserer dem i et array. Jeg kan derefter få dem vist med en while-løkke, der tæller de første 3 pladser op i arrayet. Her viser jeg de 3 nyeste spil på samme måde som på Spil-listen.

Figur : Forsiden

## Spil-liste-siden – spilListe.php

(se bilag 6)

Spil-liste-siden viser alle spil, der er blevet oprettet og sorteret efter navn, når man går ind på siden. Brugeren kan komme videre via menuen, eller vælge at sortere i listen ved at klikke på knappen ”søg”. For at søge skal brugeren vælge en kategori og pris, i boksene under menuen. Han kan også vælge knappen ”Vis alle spil”, som viser alle spille.

Figur : uddrag spil-liste-siden, (der kan være 3 bokse på en række)

Søgefunktionen fungerer via html attributten form action og input type="submit", som sender de valgte data til en side, der behandler data angivet i form action=”en-side” (I dette tilfælde er det samme side spilListe.php) Dette gøres ved følgende kode.(Se næste side)

|  |
| --- |
| <!-- søgeliste efter kategori-->  <form action=**"spilListe.php"** method=**"get"** >  **Søg 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'"**>  </form> |

Til indtastning af pris og kategori, har jeg lavet en drop-down liste. Brugeren skal ikke selv indtaste noget. Knappen søg, sender de valgte data, gemt i variablerne spilKategori og prisKategori, til samme side via GET-array’et. Fordelen er, at en bruger så kan linke en søgning af spil til en anden i stedet for at den er skjult via POST-array’et. ”Vis alle spil”-knappen opdatere siden og alle spil bliver vist da der ikke er sendt nogle værdier med GET-array’et

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='".$prisKategori."'  ORDER BY spilnavn")**or** **die**(mysql\_error());  //visning af spil  **while**($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 beskyttet mod SQL-injection og gemt i samme variabelnavne via GET-array. Herefter bliver de relevante spildata fra databasen hentet ved et SQL udtræk. Her søges efter den valgte kategori og pris med SQL koden WHERE (se $resultat i if-sætning). Så bliver de sorteret efter navn med SQL koden ORDER BY. Når dette er udført bliver de 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 så hentes fra databasen og vises.

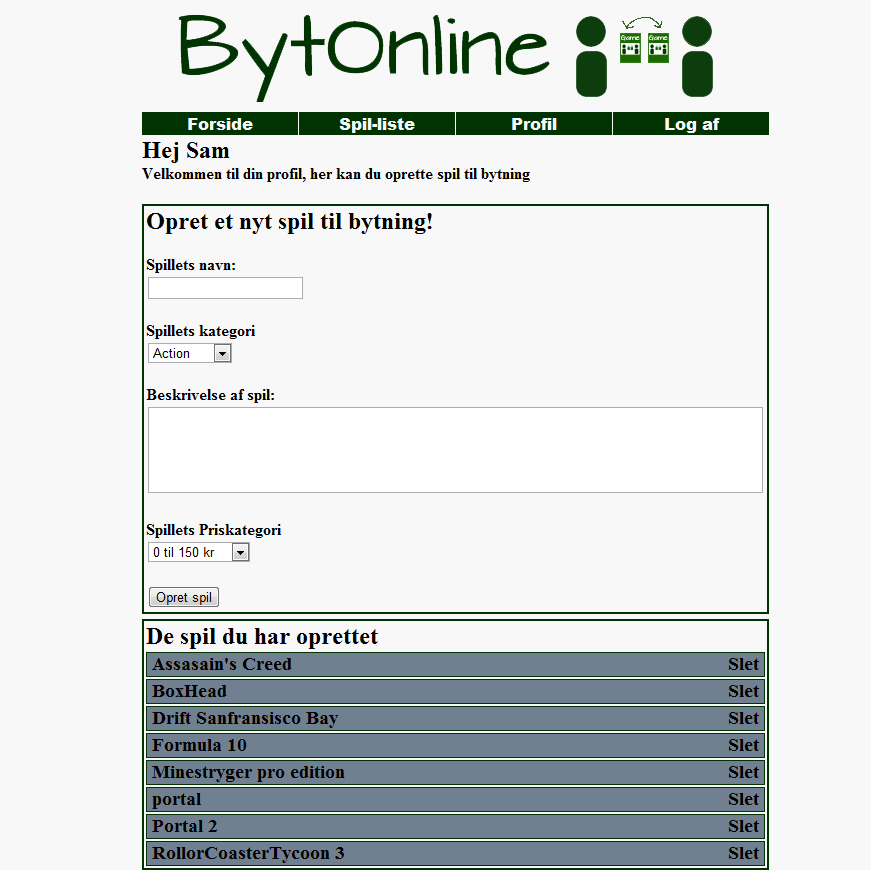
## Profil – profil.php, opretSpil.php, sletSpil.php

(se bilag 7)

Profil siden er brugerens eget sted, hvor han kan oprette spil (han ønsker at bytte), se de spil han har oprettet og slette dem. Det er også tiltænkt, at beskeder fra folk, der gerne vil bytte med ham, skal komme frem her (se videreudvikling). Hvis ikke man er logget ind, når man går ind på siden, sørger logindTjek.php for at sende en til login siden, hvorefter man bliver sendt tilbage til profil siden. Brugeren kan stadig navigere rundt mellem de andre sider i menuen.

Ved at klikke på ”opret spil” knappen, bliver de indtastede værdier tjekket og der gives en fejlbesked hvis nogle af felterne er tomme. Ellers bliver spillet oprettet og en servicbesked ”spillet er oprettet” samt tilbagelink (til profil.php) vises på opretSpil.php-siden. Klikker brugeren på slet, bliver man spurgt om man er sikker på man vil slette ved et popup boks, så man ikke kommer til at slette ved et uheld. Hvis ja, slettes spillet og lignende type servicemeddelse og link bliver vist på sletSpil.php.

Når brugeren er logget ind vises følgende.



Figur profil

Først finder jeg brugerens navn via brugerID’et som er gemt i $\_SESSION['LoggedIn']. På den måde kan jeg lave en auto velkomst. Hvilket foregår ved SQL-koden SELECT fornavn FROM brugere WHERE brugerID = "'.$\_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 input type=”submit”, og drop-down listerne er fuldstændig magen til. Her skal brugeren også indtaste overskrift og beskrivelse. Overskriften giver jeg en max længde 70 og angiver det er type=”text”, Til beskrivelse bruger html tag ”textarea” hvor jeg sætter antal række og kolonner (cols, rows). Når opret spil knappen trykkes sendes værdierne gemt i variablerne, angivet med attributten name=”navn”, 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;'  $soeg = **array**('<', '>');  $erstat = **array**('&lt;', '&gt;');    // Beskyttelse med Sanitize input  $spilNavn = mysql\_real\_escape\_string(str\_replace($soeg, $erstat, $\_POST['spilNavn']));  $spilKategori = mysql\_real\_escape\_string($\_POST['spilKategori']);  $prisKategori = mysql\_real\_escape\_string($\_POST['prisKategori']);  $spilBeskrivelse = mysql\_real\_escape\_string(str\_replace($soeg, $erstat,  $\_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 få besked 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>";  } |

Hvis alle felter er udfyldt erstattes ”<” og ”>” og der beskyttes mod sql-injection(se afsnit om sikkerhed). 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.

De oprettede spil vises som et h3 element med navn, og en slet-knap til højre. De bliver vist med nogenlunde samme kode, som til spil-listen, hvor et SQL udtræk henter brugerens spil fra databasen, hvor brugerID matcher. Slet funktionen er lavet ved et JavaScript, se koden nedenfor.

|  |
| --- |
| <!-- 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 sletSpil.php og id sendes med via GET aray’et. På sletSpil.php tjekker jeg om brugerens ID gemt i $\_SESSION['LoggedIn'] stemmer overens med det pågældende brugerID gemt i spil tabellen ud fra det valgte spilID. Dette er for at undgå at en bruger kan slette andres spil. Se koden nedenfor.

|  |
| --- |
| $spilID = mysql\_real\_escape\_string($\_GET['spilID']);  //test om man må slette spillet  $tjekBruger = "SELECT brugerID FROM spil WHERE spilID='".$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 gives beskeden at spil er slettet og brugeren kan gå til bage til profilen. Hvis brugeren ikke må slette spillet sendes man automatisk til index.php.

## Login – logind.php, logud.php

(se bilag 8)



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. Hvis man ikke har oprettet en bruger endnu kan man klikke på ”registrer dig her!” linket.

Det er en standard login form, hvor man skal skrive brugernavn og adgangskode. Klikker man log ind, tjekkes om brugernavn og adgangskode passer, samt om nogle af felterne er tomme. Der gives så en fejlmeddelelse i rød tekst øverst, hvis en af tingene er glemt. Ellers bliver man logget ind og sendt tilbage til siden man var på før. Man kan ikke gå ind på registrer bruger og log ind siderne, hvis man allerede er logget ind.

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

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. Jeg tjekker om adgangskode er forkert ved mysql\_num\_rows funktionen. Denne funktion tæller 1 op, hvis den finder det indtastede brugernavn og adgangskode i tabellen. Hvis den er mindre en 1, så sendes man tilbage til logind.php med en relevant fejlbesked. Ellers bliver værdierne beskyttet mod JavaScript og SQL-injection (se afsnit om sikkerhed) hvorefter $\_SESSION['LoggedIn'] bliver sat til at være brugerens ID som jeg henter ved et SQL udtræk fra databasen. Brugeren er nu logget ind.

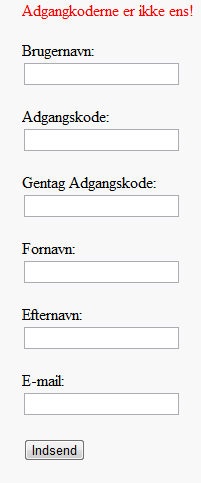
Bemærk at jeg krypterer koden med hash-kryptering. Hash-kryptering vil altid give den samme værdi for den samme adgangskode, og jeg kan derfor tjekke med databasen om brugeren har skrevet den rigtige adgangskode. (se evt. afsnit om sikkerhed). Dette er fordi at brugerens adgangskode under oprettelse og bliver hashkrypteret.

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

# Registrerer – registrer.php, registrerQuery.php

(se bilag 9)

Trykker man på registrer fra logind.php siden, 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 input type”submit” knap der går til siden registrerQuery.php. På billedet viser jeg kun formen, men menuen øverst er der også.



Figur registrer form

Når brugeren har udfyldt felterne, sørger jeg for at kryptere adgangskoden, og beskytter databasen mod SQL-injection samt JavaScript. Jeg tjekker også for en masse brugerfejl, som f. eks om adgangskoderne er forskellige. Jeg bruger også samme fejlsystem til log ind og oprettelse af spil. 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. Fejlsystemet er bygget op 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 tester f. eks om brugeren allerede er oprettet ved koden nedenfor. Hvis en bruger allerede eksisterer af samme navn sender jeg fejlkoden 3 tilbage til registrer.php og den vises øverst.

|  |
| --- |
| $tjekBrugernavn = "SELECT brugernavn FROM brugere WHERE brugernavn='".$brugernavn."';";  $tjek = mysql\_query($tjekBrugernavn) **or** **die**(mysql\_error());  **if**(mysql\_num\_rows($tjek) > 0) {  header('Location: registrer.php?error=3&redirect='.$\_GET['redirect']);  **exit**;  } |

Hvis der ikke er fejl gemmer jeg brugerens data i brugertabellen med en række relevante SQL-quries, og brugeren er nu oprettet. Dette foregår nogenlunde på samme måde som ved oprettelse af spil.

## Test af hjemmesiden

Jeg har løbende 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. F. eks er det en god ide at tjekke at de data man hiver ud af databasen er de rigtige værdier, ved at udskrive dem før man begynder at bruge dem i noget andet kode. I næste afsnit ridser jeg nogle kendte fejl op, som jeg ikke har løst endnu.

### Kendte fejl/mangler

Teksten der bliver vist i spilboksene, 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 drop down boksene altid viser den samme priskategori og spilkategori når man har trykket søg, i stedet for at vise det man søgte efter.

Hvis man kun skriver mellemrum i nogle felter, så kan det se underligt ud med CSS’en. F. eks hvis der oprettes et spil med mellemrum og intet andet, som vil blive godkendt, da mellemrum ikke tæller for et tomt felt.

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

spilID bliver vist i URL’en via JavaScript’et, når man vil slette et spil. Det gør at man kan indskrive spilID selv. Man kan dog kun slette sine egne spil, så det går ikke udover andre.

### Hyppige fejl

Jeg har haft flest fejl omkring syntakserne fordi det kan være svært at holde styr på kode, når man går fra HTML til PHP til MySQL på den ene eller anden måde! Ellers har jeg lavet små fejl, som at glemme at opdatere variabel navne ved ændringer og fejl når man skulle sende værdier med GET-arrayet. Efterhånden som jeg er blevet mere øvet med serverprogrammering, laver jeg også mindre fejl

### Bruger testen

Jeg har med vilje forsøgt at gør alt det forkerte, for at se om en bruger har kunnet smutte uden om log ind krævede sider samt forsøgt i alle felter 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 kun skrev mellemrum i nogle felter, så virker noget af CSS koden ikke virker, og det ser underligt ud.

Ellers burde jeg have dækket de fleste sikkerhedshuller omkring ikke ondsindede brugerfejl. (se evt. afsnit om sikkerhed)

# Endelige sluttest

Jeg har fået nogle som ikke har hørt hvad min hjemmeside går ud på, til at teste den. Dette er for at se om hjemmesiden umiddelbart giver mening, forstået på den måde at brugerne gør som jeg har tiltænkt de skal. Jeg lavede en kvalitativ analyse, med 2 personer, en mand og en kvinde.

Resultaterne var fine. Da jeg bagefter spurgte hvad de mente hjemmesiden gik ud på, var de ikke i tvivl om at man kan bytte spil. Jeg fik lidt forslag til tekstændringer et par steder. Der var ellers ingen problemer, de gjorde som de skulle med at oprette både bruger og spil bagefter. Da selve bytningssystemet ikke er lavet (i dette projekt), spurgte begge til hvordan de skulle bytte spil. Så det var tydeligt, at de forstod hvad hjemmesiden går ud på.

# Sikkerhed

Jeg har sørget for at beskytte min database mod sikkerheds huller som cross-scripting, SQL-injection, og GET-array’s (hvor bruger f. eks kan indtaste ondsindede kode i URL). Udover dette sørger jeg for at hash-krypteret adgangskoderne.

mysql\_real\_escape\_string() funktionen bruger jeg i alle SQL-statements hvor brugeren har mulighed for at indtaste værdier. Både gennem GET-array’et og på de forskellige sider som ”form action=”en-side.php”” fører en hen til. På den måde undgår jeg, at brugeren kan skrive statements som ’;drop table spil; i et SQL-udtræk fra databasen(som ville slette spil tabellen), fordi jeg escaper visse symboler og tegn. SQL-injection kan ud over at slette en hel tabel f. eks også trække adgangskoder og brugernavne ud af databaser, eller gøre hackeren til administrator[[7]](#footnote-7).

Hvis det skulle lykkedes at hente adgangskoder ud af den database, vil den ondsindede hacker ikke få noget ud af det. Det er fordi adgangskoderne er hash-krypteret. Det smarte ved en hash-funktion, som er en algoritme, er, at den tager en kort eller lang besked og omdanner den til en hash-værdi af en bestemt størrelse. Man kan aldrig genskabe den oprindelige besked, men man kan køre den samme hashalgoritme på f. eks en adgangskode skrevet 2 steder og sammenligne de 2 hash-værdier. Hash værdien er nemlig altid den samme hvis samme algoritme er brugt på samme besked. Derfor hashkrypterer jeg også adgangskoder i min database og kan på den måde sikre mig at ingen kan læse adgangskoden i selve databasen. Hash-kryptering beskytter altså kun mod hackere, der får uautoriseret adgang til databasen. Jeg bruger en PHP hash funktion kaldet md5.

Jeg sørger også for at erstatte ”<” og ”>” tegnene, så der ikke kan skrives ondsindet JavaScripts, eller så en bruger kan bruge irriterende tags (f. eks skrive navn med fed). JavaScript kan f. eks bruges til at ændre værdierne i en html form[[8]](#footnote-8) hvilket kan gøres i f. eks en hidden form, der måske ikke er beskyttet mod SQL-injection – for så er der et problem.

# 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. Dette skal foregå på bytternes profil, hvor de kan sende beskeder til hinanden. Ellers havde jeg tænkt mig, 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 kan også gøres til links, så man kan klikke på pågældende spil man vil bytte med. Endvidere kan spil-liste sidens søge funktioner godt gøres mere avanceret.

# Konklusion

BytOnline er endt med at blive en funktionel og brugervenlig hjemmeside, som opfylder de mål jeg satte mig fra starten, endda med lidt ekstra funktionalitet. Skrevet i XHTML, JavaScript, PHP, MySQl samt CSS.

Jeg har fokuseret på problemet om at nye computerspil er dyre at anskaffe sig, og at de gamle/gennemførte ikke længere bliver spillet. Løsningen har jeg set i at man kan bytte computerspil med hinanden, både for at spare penge og for morskabens skyld.

Jeg har planlagt min hjemmeside med E/R diagram og fået min database på 3 normalform. Jeg har lavet funktionalitet som f. eks oprettelse af database, oprettelse af bruger og log ind samt oprettelse af spil og spilliste med sorteringsfunktion. Jeg har også lavet et layout til brugerinterfacet og kodet mig ud af nogle sikkerhedshuller, samt beskyttet mod nogle former for hacking. (Cross scripting og SQL-injection)

Jeg har afslutningsvis testet min hjemmeside og fået positive tilbagemeldinger. Blandt andet at hjemmesidens formål var tydeligt, og funktionerne nemme at forstå og bruge.

# Bilag

## Ophavsret

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

## Kode

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

|  |
| --- |
| <!--For at kunne installere databasen kræves det at man conecter til phpmyadmin med brugernavn og password-->  <span style=**"color:red"**>**Skriv brugernavn og password for at installere databasen i systemet**</span><br /><br />  <form method=**"post"** action=**"config.php"**>  **MySQL Username**<br />  <input type=**"text"** name=**"mysql\_username"** /><br />  **MySQL Password**<br />  <input type=**"password"** name=**"mysql\_pw"** /><br />  <input type=**"submit"** value=**"Installer"** name=**"submitted"** />  </form> |

|  |
| --- |
| <!--  Denne fil skaber databasen, bliver kørt gennem installer.php  -->  <?php  //Får at connecte til databasen så forventer den et brugernavn og et password  **if**(!**isset**($\_POST['mysql\_username']) **or** !**isset**($\_POST['mysql\_pw'])){ //sørger for at unøskede ikke kan rode ved oprettelse af databasen  **echo**'<span style="color:red">INGEN ADGANG</span>';  **exit**;  }  // conecter til phpMyAdmin  **if**($conn = mysql\_connect('localhost',$\_POST['mysql\_username'],$\_POST['mysql\_pw'])) {  **echo** 'Logget ind!<br />';  **echo** 'Tjekker databasen bytOnline<br />';  // 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 />';  }  /\*sikkerhedsbruger  Dette den her kode gør er at den skaber privilegier til bytOnline databasen, men ik ke nogen andre databaser. På den måde undgås det at det folk udefra kan pille ved andre databaser hvis man skulle glemme at beskytte sig mod indtrængen  \*/    $tjekBruger = "SELECT user FROM mysql.user WHERE user='eksamensprojekt'";  **if**(mysql\_num\_rows(mysql\_query($tjekBruger)) < 1) {//tjekker om brugeren er oprettet fr  $query = "CREATE USER 'eksamensprojekt'@'localhost' IDENTIFIED BY  'kummefryser';";  $grant = "GRANT ALL PRIVILEGES ON `bytOnline` . \* TO 'eksamensprojekt'@'  localhost'WITH GRANT OPTION ;";  mysql\_query($query) **or** **die**(mysql\_error());  mysql\_query($grant) **or** **die**(mysql\_error());  **echo** "eksamensprojekt bruger oprettet <br />";  }    //brugere tabel  $tjekBrugere = "SELECT \* FROM brugere;";  $resultat = mysql\_query($tjekBrugere);  **echo** 'Tjekker brugere tabellen <br />';  **if**(!$resultat) {  **echo** 'Laver brugertabellen<br />'; //bruger, fornavn, efternavn og email er 4 gange så stor så '<' and '>' kan erstattes med '&lt;' og '&gt;'  $lavBrugere = " CREATE TABLE `bytOnline`.`brugere` (  `brugerID` INT( 10 ) NOT NULL AUTO\_INCREMENT ,  `brugernavn` VARCHAR( 64 ) NOT NULL ,  `password` VARCHAR( 50 ) NOT NULL ,  `fornavn` VARCHAR( 120 ) NOT NULL ,  `efternavn` VARCHAR( 120 ) NOT NULL ,  `email` VARCHAR( 120 ) NOT NULL ,  `isAdmin` TINYINT( 1 ) NOT NULL DEFAULT '0' ,  PRIMARY KEY ( `brugerID` ) ,  UNIQUE (  `brugernavn` ,  `email`  )  ) ENGINE = INNODB;";  mysql\_query($lavBrugere) **or** **die**(mysql\_error());  **echo** 'Brugere tabel oprettet <br />';  } **else**{  **echo** 'Brugere tabellen er allerede oprettet <br />';  }    //spil tabel  $tjekSpil = "SELECT \* FROM spil;";  $resultat = mysql\_query($tjekSpil);  **echo** 'Tjekker spil tabel <br />';  **if**(!$resultat) {  **echo** 'Laver spiltabellen<br />';  $lavSpil = " CREATE TABLE `bytOnline`.`spil` (  `spilID` INT( 10 ) NOT NULL AUTO\_INCREMENT PRIMARY KEY ,  `brugerID` INT( 10 ) NOT NULL,  `spilnavn` VARCHAR( 280 ) NOT NULL ,  `spilbeskrivelse` TEXT NOT NULL ,  `spilkategori` VARCHAR( 30 ) NOT NULL ,  `priskategori` VARCHAR( 30 ) NOT NULL  ) ENGINE = INNODB;";  mysql\_query($lavSpil) **or** **die**(mysql\_error());  **echo** 'Spil tabel oprettet <br />';  } **else** {  **echo** 'Spil tabel allerede oprettet <br />';  }    // bud tabel  $tjekBud = "SELECT \* FROM bud;";  $resultat = mysql\_query($tjekBud);  **echo** 'Tjekker bud tabel<br />';  **if**(!$resultat) {  **echo** 'Laver budtabellen<br />';  $lavBud = " CREATE TABLE `bytOnline`.`bud` (  `budID` INT( 10 ) NOT NULL AUTO\_INCREMENT PRIMARY KEY ,  `besked` TEXT NOT NULL ,  `selgerID` INT(10) NOT NULL ,  `koberID` INT(10) NOT NULL  ) ENGINE = INNODB;";  mysql\_query($lavBud) **or** **die**(mysql\_error());  **echo** 'Bud tabel oprettet <br />';  } **else** {  **echo** 'Bud tabel allerede oprettet <br />';  }  //Relationer  **echo** 'laver relationer <br />';    //Spil  $brugereRelation = "ALTER TABLE `spil` ADD INDEX ( `brugerID` )";  $lavRelationForBruger = "ALTER TABLE `spil` ADD FOREIGN KEY ( `brugerID` )  REFERENCES `bytOnline`.`brugere` (  `brugerID`  ) ON DELETE CASCADE ON UPDATE CASCADE ;";  mysql\_query($brugereRelation) **or** **die**(mysql\_error());  mysql\_query($lavRelationForBruger) **or** **die**(mysql\_error());    //Bud  $budSelgerRelation = "ALTER TABLE `bud` ADD INDEX ( `selgerID` )";  $budKoberRelation = "ALTER TABLE `bud` ADD INDEX ( `koberID` )";  mysql\_query($budSelgerRelation) **or** **die**(mysql\_error());  mysql\_query($budKoberRelation) **or** **die**(mysql\_error());    $lavRelationForSelgerBud = "ALTER TABLE `bud` ADD FOREIGN KEY ( `selgerID` )  REFERENCES `bytOnline`.`spil` (  `spilID`  ) ON DELETE CASCADE ON UPDATE CASCADE ;";  mysql\_query($lavRelationForSelgerBud) **or** **die**(mysql\_error());    $lavRelationForKoberBud = "ALTER TABLE `bud` ADD FOREIGN KEY ( `koberID` )  REFERENCES `bytonline`.`spil` (  `spilID`  ) ON DELETE CASCADE ON UPDATE CASCADE;";  mysql\_query($lavRelationForKoberBud) **or** **die**(mysql\_error());    **echo** 'relationer er oprettet<br />';  }  ?>  <br /><a href=**"index.php"**>**Return to Index page**</a><!-- tilbage til forside link--> |

### Bilag 2 stylesheet.css

|  |
| --- |
| /\* Cascadig style sheets (css)\*/  body**{**  **padding: 0;**  **margin: 0;**  **background-color:#F8F8F8;**  **}**  /\* denne division srger for at  alt content vil blive centrerert i midten\*/  div**.**content **{**  **width: 640px;**  **margin-left: auto;**  **margin-right: auto;**  **}**  /\* denne division srger for at  aligne elementerne inden for  content og menuebaren så det  ser pænere ud\*/  **.**justering**{**  **margin-top: 0px;**  **margin-bottom: 0;**  **margin-left: 0.8%;**  **margin-right: 1.3%;**  **overflow: hidden;**  **}**  /\*css til menu-linjen \*/  **.**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;**  **}**  **.**mulighedsboks **{** /\*Boksen rundt om sortering af spil på spil-listesiden\*/  **width: 619px;**  **height: auto;**  **border-width: 2px;**  **border-style: solid;**  **border-color: #003300;**  **float: left;**  **margin-left: 0.8%;**  **margin-right:1px;**  **overflow: hidden;**  **padding: 2px;**  **}**  **.**spilboks **{**/\*Boksen rundt om spil på spil-listesiden \*/  **width: 201.33333333333333333333333333333px;**  **height: 250px;**  **border-width: 2px;**  **border-style: solid;**  **border-color: #003300;**  **border-bottom: 3px solid #003300;**  **float: left;**  **margin-left:0.8%;**  **margin-right: 1px;**  **overflow: hidden;**  **margin-top: 5px;**    **}**  **.**egnespil**{** /\* listen over egne spil i profil.php\*/  **width: 619px;**  **height: auto;**  **border-width: 2px;**  **border-style: solid;**  **border-color: #003300;**  **float: left;**  **margin-left: 0.8%;**  **margin-right:1px;**  **overflow: hidden;**  **padding: 2px;**  **margin-top: 5px;**  word-wrap**:break-word;**  **}**  h2 **{** /\* fjerne webbrowserens egen indryk og alignment\*/  **padding: 0;**  **margin: 0;**  **}**  h3 **{** /\* fjerne webbrowserens egen indryk og alignment\*/  **padding: 0;**  **margin: 0;**  **}**  **.**egnespil h3 **{** /\* listen over egnespil på profil.php\*/  **background-color: #708090;**  **color: #000000;**  **margin-top: 2px;**  **margin-bottom: 0;**  **border-width: 1px;**  **border-style: solid;**  **border-color: #003300;**  word-wrap**:break-word;**  **}**  **.**spilboks h3 **{** /\*overskriften i spilboksene \*/  **background-color: #00e57c;**  **color: #000000;**  **margin-top: 0;**  **margin-bottom: 0;**  **overflow: hidden;**  **}**  **.**spilboks h3 span **{** /\* justering af overskrift i spilboks\*/  **display: block;**  **margin-left: 5px;**  **margin-right: 5px;**  word-wrap**: break-word;**  **}**  **.**spilboks p **{**/\* justering af bekskrivelse af spil i spilboks\*/  **margin-top: 1px;**  **padding:0;**  **margin-left: 5px;**  **margin-right: 5px;**  word-wrap**:break-word;**  **}** |

### Bilag 3 – util.php

|  |
| --- |
|  |

1. Fra ”introduktion til serverprogrammering” dokument udleveret af læreren. [↑](#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 side 51 [↑](#footnote-ref-5)
6. Billedet er fra <http://www.onlinemovieshut.com/wp-content/uploads/2010/12/uncharted-drakes-fortune.jpg> [↑](#footnote-ref-6)
7. Iføgle ”PHP og MySQL” bog af Kristian Langborg-Hansen side 65 [↑](#footnote-ref-7)
8. <http://www.testingsecurity.com/how-to-test/injection-vulnerabilities/Javascript-Injection> [↑](#footnote-ref-8)