ProgeTiigri õppematerjal

Serveripoolsete veebirakenduste koostamine

Jaagup Kippar

2012

Sisukord

Sissejuhatus

Märgatav osa tänapäevastest arvutirakendustest töötab veebis. Mõned neist kasutavad veebiühendust vaid lehe kohale tõmbamiseks ning edasine töö käib kliendi arvutis või mobiilseadmes. Sellisel juhul piisab veebiserveriks lihtsast failide laust, kust need küsimise peale siis kasutaja arvutisse saadetakse. Kui aga soovitakse rakenduses loodud andmeid mõne teise arvutiga vahetada, siis peab veebiserver veidi targem olema. Vahetamist on tarvis teisele kasutajale andmete näitamiseks, samuti kui soovitakse ise oma töö talletada ja hiljem teises arvutis edasi teha. Suuremate andmekoguste edasikandmise tarbeks on olemas ka mälupulgad ja muud salvestusseadmed. Kuni aga õnnestub sekundite jooksul veebi kaudu andmed sobivasse kohta saada, siis see on päris mugav.

Veebilehtedele sai serveripoolset tarkust jagada juba 1990ndate algul. CGI-nimelise liidese kaudu võib igas programmeerimiskeeles kirjutatud programm saada veebist andmeid ning saata oma tulemuse kasutajale. Nõnda olid algselt veebilehedki kirjutatud C-s, Pascalis, PERLis ja muudes parasjagu levinud keeltes. Veebi levides asuti programmeerimiskeeltele veebisõbralikke täiendusi looma ning veebi jaoks suisa eraldi keeli ja tehnoloogiaid kokku panema. Üheks esimeseks selliseks sai ka PHP, mille põhjal siinne õpik koostatud. PHPd kasutatakse pigem väikeste ja keskmise suurustega veebide loomisel ning Eestis töötavate veebide üldarvust on vähemalt kaks kolmandikku PHPga seotud - olgu siis otse selle keele abil tehtud või kasutatud mõnd raamistikku, mis PHPle tugineb. Suuremate veebide loomise puhul konkureerivad tehnoloogiatena Java J2EE koos Servlettide ja JSP-lehtedega ning Microsoft .NETi alla kuuluv ASP.NET raamistik. Kuid ka PHP juurde on loodud vahendeid, mis aitavad lahendustel töötada tuhandete kasutajate ja sadade üheaegsete päringutega.

Veebisisendi ja -väljundiga programmidel on aga mõned eripärad. Olenevalt rakendusest, kuid küllalt sageli tuleb väljastada suures koguses muutumatut HTML-teksti ning sinna vahele vaid üksikud kohad, mis programmiga muuta vaja. Teiseks probleemiks veebirakenduste juures on, et kunagi ei saa usaldada sisendit kasutajalt – üle veebi võib tulla ligi suvaline häkker ning otsisõna asemele panna teele näiteks mõne videofilmi sisu binaarkujul. Sekelduste vältimiseks on seetõttu kasulik lisada sisendile piiranguid ja kontrolle.

Algus

Vahepealsetest keerdkäikudest hoolimata paistab, et veebi koostamisel langeb põhirõhk lähiaastatel ikka HTMLi, sinna juurde kuuluva CSSi ja Javaskripti peale. HTMLi kirjutamise ning sealtkaudu saadavate andmetega saab PHP igati mugavasti hakkama. Seetõttu alustamegi õppematerjali lihtsa veebilehe loomisega ning vaikselt lisame sinna programmeerimisvõimalusi.

Tutvustav veebileht

HTML-faili tüübiks on tekstifail. See tähendab, et see fail võib sisaldada ainult teksti või muid ekraanil nähtavaid sümboleid. Tavalise tekstina olev osa paistab jutuna kasutajatele. Teksti osade eristamiseks aga saab teksti sisse panna < ja > märkide vahele HTMLi märgendeid, millega veebilehitsejat juhatada ja dokumendi struktuuri märkida.

Veebilehe esmaseks struktuuriks on deklaratsioon (`doctype`), HTML-dokument ise - mis jääb märgendite <html> ja </html> vahele. Ning dokument jaguneb päiseks (`head`) ning sisuosaks (`body`). Esimeses neist on andmed dokumendi kohta - kes tegi, millal tegi, kui kaua kehtib, märksõnad, lühikirjeldus jm. Sisuosas (`body`) olev tekst on üldiselt vaatajale nähtav - välja arvatud märgendite nimed, millega teksti juhitakse.

```
<!doctype html>
<html>
  <head>

  </head>
  <body>

  </body>
</html>
```

Head-osas paiknev `title` on näha veebilehitseja tiitliribal. Kui korraga avatud palju lehti, siis selle järgi saab vaadata, millist lehte parajasti näha soovitakse.

Sisus võib teksti jagada lõikudesse `div`-märgendite abil. Reavahetust tähistab `<br />`. Pealkirja märkimiseks sobivad `<h1>`koos lõpuga `</h1>` ning `<h2>` ja `</h2>`. Sügavama struktuuri puhul ka suuremate numbritega pealkirjad.

Loetelu piiritleb `<ul>` ja `</ul>` (unordered list). Iga loeteluelemendi ümber tuleb `<li>` ja `</li>`. Selliste vahenditega saabki lihtsa veebilehe kokku.

```
<!doctype html>
<html>
  <head>
     <title>Jaagupi leht</title>
  </head>
  <body>
     <h1>Jaagup</h1>
       <div>
          Sündis 03. mail 1976
              Tallinnas. <br /> Kasvas Mustamäel.
       </div>
       <h2>Huvialad</h2>
       <ul>
```

```
        <li>Kandlemäng</li>
        <li>Jalgrattasõit</li>
      </ul>
  </body>
</html>
```
Veebileht nähtuna brauseriaknas:

PHP algus

Eelnev leht saadeti serverist veebilehitsejasse samal kujul nagu ta faillina oli salvestatud. Serveripoolse rakenduse korral aga pannakse failis olev kood enne käima ning siis saadetakse töö tulemus brauserisse. PHP puhul tuleb käivitatav kood panna algustähise <?php ning lõpu ?> vahele. Käsklus `echo` trükib tulemuse ekraanile

```
Arvutus:
<?php
  echo 3+2;
?>
```

Väljund lehel

Arvutus: 5

Ülesandeid

* Hangi või tee selgeks enesele võimalus PHP-võimelises veebiserveris veebilehtede loomiseks. Windows-masinas sobib näiteks XAMPP või WAMP-nimeline komplekt.
* Koosta tervitav leht ja vaata seda veebiserveri kaudu
* Muuda lehe sisu ning uuendusnupu vajutuse järel veendu muutuse kajastumises ka veebilehitsejas.
* Käivita konspektis olnud näide kahe arvu liitmise kohta.
* Muuda arve ja tehet, kontrolli tulemusi.

Muutuja, tingimuslause

Programmeerimiskeeltes on levinud võimalus andmeid muutuja ehk märksõna alla meelde jätta.

PHPs algavad muutujate nimed dollarimärgiga. See võimaldab neid hiljem vabamalt teksti sisse panna. Lõik

```
  $eesnimi="Juku";
  echo "Tere, $eesnimi!";
```

trükib aimatavalt välja ”Tere, Juku”. Samuti siis

```
  $eesnimi="Juku";
  echo "$eesnimi tuli hommikul kooli.<br />";
```

teatab, kes hommikul kooli tuli. Reavahetus `<br />` lause lõpus hoolitseb, et järgnevat juttu alataks järgmiselt realt.

Valiku jaoks on käsklus if. Tingimus pannakse ümarsulgude sisse. Kui tingimus vastab tõele (praegusel juhul kinganumber on väiksem kui nelikümmend), siis täidetakse järgnevate looksulgude vahele paigutatud plokk. Sama lugu ka alumise tingimusega.

```
<?php
  $eesnimi="Juku";
  echo "$eesnimi tuli hommikul kooli.<br />";

  $kinganumber=37;
  if($kinganumber<40){
     echo "$eesnimi saab veel lasteosakonnast kingi<br />";
  }
  if($kinganumber>45){
     echo "Kodanik $eesnimi kingad sobivad naelakastideks.";
  }
?>
```

Väljund ekraanil:

Juku tuli hommikul kooli.
Juku saab veel lasteosakonnast kingi

Ülesandeid

- Lisa muutuja perekonnanime tarbeks. Anna sinna väärtus ja kasuta seda lausetes.
- Koosta eraldi leht, kus kirjas temperatuur. Väljasta, kas vesi on sellel temperatuuril vedel või muutub jääks.

Kellaaeg

Igal lehe avamisel näitamine on kindel kontroll selle kohta, et veebiserveris lehe loomiseks midagi toimub. Kuna samast serverist võidakse lehti vaadata mitmelt poolt, siis on tarvilik teada anda, millise ajavööndi kellaaega soovitakse. Edasi juba käsklus date annab soovitud tulemuse. Sulgudes olevad tähed näitavad, kuidas kellaaega vormistada. Tekst ikka jutumärkide vahel. Suur H palub tunde näidata 24-tunni süsteemis, i tähendab minuteid ning s sekundeid. Koolonid seal vahel trükitakse lihtsalt välja.

```
<!doctype html>
<html>
  <head>
```

```
    <title>Kellaleht</title>
  </head>
  <body>
    <h1>Kellaaeg</h1>
      <?php
        date_default_timezone_set("Europe/Tallinn");
        echo date("H:i:s");
      ?>
  </body>
</html>
```

Väljund:

18:31:39

Ülesandeid

- Näita kellaajast vaid tunde ja minuteid
- Näita välja kuupäev. `Y` tähistab aastat, `m` kuud ning `d` päeva.

Tollikalkulaator

Nagu arvuti sõnastki välja lugeda võib, on arvutamine selle masina juures tähtis ülesanne. Vahel saab arvutada olemasolevate andmete põhjal. Küllalt sageli aga on vaja kasutajalt algandmete määramiseks sisestust. Andmete sisestamiseks on veebilehe jaoks olemas vorm ehk sisestuselementide komplekt. Form-elemendi action-atribuudina määratakse, kuhu aadressile andmed töötlemiseks saadetakse. Kui sihtkohaks on küsimärk, siis tuleb sisestus samale aadressile, kus vormgi avanes.

```
<!doctype html>
<html>
  <head>
    <title>Arvutamine</title>
  </head>
  <body>
    <h1>Tollikalkulaator</h1>
    <form action="?">
          Monitori diagonaal tollides:
              <input type="text" name="tollid" />
              <input type="submit" value="OK"  />
    </form>
  </body>
</html>
```

Tollikalkulaator

Monitori diagonaal tollides: OK

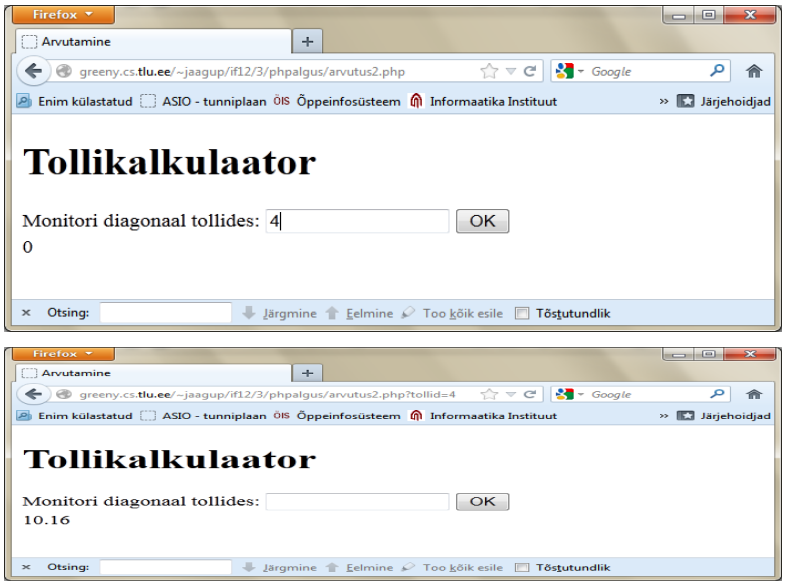
Andmete kasutamiseks tuleb nad kinni püüda. Muutuja $_REQUEST kaudu saab vormi sisestatud väärtused kätte.

```
echo $_REQUEST["tollid"]*2.54;
```

korrutab saabunud tollide arvu 2,54ga ning väljastab tulemuse sentimeetrites.

```
<!doctype html>
<html>
  <head>
     <title>Arvutamine</title>
  </head>
  <body>
     <h1>Tollikalkulaator</h1>
     <form action="?">
           Monitori diagonaal tollides:
               <input type="text" name="tollid" />
               <input type="submit" value="OK"  />
       </form>
       <?php
          echo $_REQUEST["tollid"]*2.54;
       ?>
  </body>
</html>
```

Tulemus näha pildina:



Eelmine näide oli küll võimalikult lihtne, aga mõnigate puudustega. Lehe avamisel ilmub sinna salapärane ümmargune null. Vastavalt serveri seadetele raskemal juhul isegi veateade andmete puudumise kohta. Seega on hea leht viisakamaks teha. Aitab järgmine lõik:

```
if(empty($_REQUEST["tollid"])){
  echo "Ootan sisestust.";
} else {
  echo $_REQUEST["tollid"]." tolli on ".
       ($_REQUEST["tollid"]*2.54)." cm.";
}
```

Inimkeelne tõlge. Kui saabuvaid tolle pole, siis trükitakse, et "Ootan sisestust.". Muul juhul trükitakse, et mitu tolli on mitu sentimeetrit. Kusjuures empty on selline käsklus, mis loeb tühjaks nii parameetri täieliku puudumise (näiteks lehe esmakordsel avamisel, kus keegi ei teagi tolle sisestada) kui lihtsalt tühjaks jäänud teksti (mis juhtub siis, kui vajutada OK-nupule ilma tollide

arvu sisestamata). Leht tervikuna.

```
<!doctype html>
<html>
  <head>
     <title>Arvutamine</title>
  </head>
  <body>
     <h1>Tollikalkulaator</h1>
     <form action="?">
           Monitori diagonaal tollides:
               <input type="text" name="tollid" />
               <input type="submit" value="OK"  />
       </form>
       <?php
         if(empty($_REQUEST["tollid"])){
           echo "Ootan sisestust.";
         } else {
           echo $_REQUEST["tollid"]." tolli on ".
                ($_REQUEST["tollid"]*2.54)." cm.";
         }
       ?>
  </body>
</html>
```

Vastuses siis kõigepealt oodatakse sisestust.

Hiljem antakse viisakas vastus.

Ülesandeid

- Koosta kalkulaator sentimeetritest tollide arvutamiseks. Jagamismärgiks on kaldkriips /.
- Koosta kalkulaator, kus sisestus on meetrites. Väljastatakse, mitu kilomeetrit see on, mitu detsimeetrit, mitu sentimeetrit ning mitu millimeetrit see on.

Ostusumma arvutamine

Sisestatavaid väärtusi võib olla mitu. Nende kättesaamiseks tuleb nad panna eri nimedega tekstiväljadesse. Järgnevas näites on ühe välja nimeks `hind` ja teise nimeks `kogus`. Neid kahte

korrutades saadakse kokku ostusumma.

```
<!doctype html>
<html>
  <head>
     <title>Arvutamine</title>
  </head>
  <body>
     <h1>Summa kalkulaator</h1>
     <form action="?">
Kauba tükihind:
               <input type="text" name="hind" />
Ostetav kogus:
               <input type="text" name="kogus" />
               <input type="submit" value="OK"  />
       </form>
       <?php
         if(empty($_REQUEST["hind"]) or empty($_REQUEST["kogus"])){
           echo "Ootan sisestust.";
         } else {
           echo $_REQUEST["hind"]*$_REQUEST["kogus"];
         }
       ?>
  </body>
</html>
```

Valik rippmenüüst

Tekstiväli on sisestuseks mugav teha. Kuna aga kasutaja pääseb sinna kõike sisestama, siis võib kergesti sisse sattuda ka sobimatut teksti. Kasutajale piiratud arvu valikute andmiseks sobib rippmenüü. Rippmenüü loob element tüübist select. Elemendi nime järgi saab endiselt PHP kaudu kasutaja sisestuse kätte.

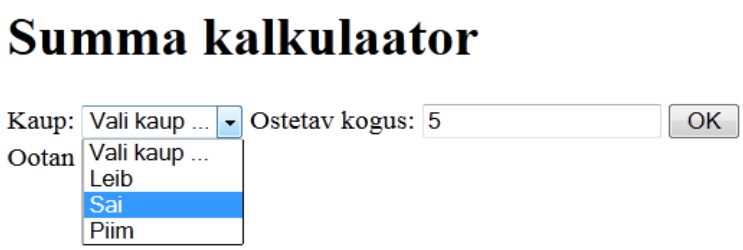
```
       <select name="hind">
            <option value="">Vali kaup ...</option>
            <option value="0.70">Leib</option>
            <option value="0.60">Sai</option>
            <option value="0.50">Piim</option>
         </select>
```

Atribuudi value juurde kirjutatud väärtus läheb serverisse edasiseks töötluseks, tekst enne option-elemendi lõppu jääb näha kasutajale oma valiku tegemisel. Allpool saab arvutada endist moodi.

```
<!doctype html>
<html>
  <head>
     <title>Arvutamine</title>
  </head>
  <body>
     <h1>Summa kalkulaator</h1>
     <form action="?">
```

```
Kaup:
       <select name="hind">
            <option value="">Vali kaup ...</option>
            <option value="0.70">Leib</option>
            <option value="0.60">Sai</option>
            <option value="0.50">Piim</option>
         </select>
Ostetav kogus:
               <input type="text" name="kogus" />
               <input type="submit" value="OK"  />
       </form>
       <?php
         if(empty($_REQUEST["hind"]) or empty($_REQUEST["kogus"])){
           echo "Ootan sisestust.";
         } else {
           echo $_REQUEST["hind"]*$_REQUEST["kogus"];
         }
       ?>
  </body>
</html>
```

Tulemus lehel:



Ülesandeid

- Pane näide tööle
- Muuda tooteid ja hindu, katseta tulemust.
- Loo toodetest kaks rippmenüüd eri nimedega. Näita, kui suur summa tuleb kummagi toote kohta eraldi ning kui palju kahe toote peale kokku.

Andmed massiivist

Paar harva muutuvat väärtust on hea veebilehe sisse kirjutada. Kui aga andmeid rohkem, neid kasutatakse mitmes kohas või nad kipuvad sageli muutuma, siis on tavaline, et andmete kirjapaneku ja kasutamise kohad erinevad. Andmed on mugav kirja panna eraldi andmebaasi või eraldi faili. Et seda pole veel õpitud, siis piirdume ühise massiiviga, mida ka vajadusel mitmel pool kasutada saab.

Massiivi loomiseks sobib käsklus array(). Edasi võib sinna ükshaaval andmed sisse panna. PHP lubab massiivi võtmena kasutada ka teksti. Kui kirjutatakse $kaubad["vorst"]="2.50", siis muutujaks on $kaubad, võtmeks "vorst" ning väärtuseks "2.50".

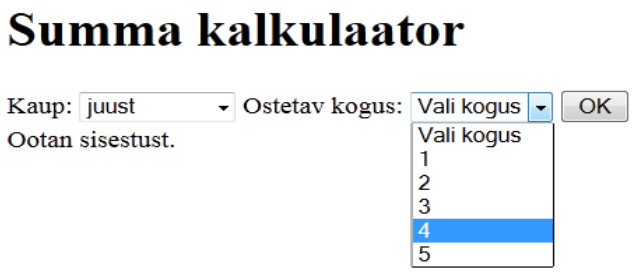
```
$kaubad=array();
$kaubad["vorst"]="2.50";
$kaubad["juust"]="3.00";
$kaubad["kartul"]="0.45";
```

Pärast võimalik tsükliga andmed läbi käia. Käsk foreach võtab ükshaaval ette kõik võtme ja väärtuse paarid (mis siinsel juhul on $nimetus ja $hind) ning lubab nendega tsükli keha sees (ehk looksulgude vahel) toimetada. Tulemusena trükitakse välja kõik valikud nõnda, et nimetused jäävad kasutajale silma ette valida. Hinnad aga saadetakse pärast valiku tegemist ning sisestusnupule vajutamist serverisse.

```
          foreach($kaubad as $nimetus => $hind){
                echo "<option value='$hind'>$nimetus</option>";
          }
```

Serveris tulemuse kokku arvutamine käib ikka endisel moel.

```
<?php
   $kaubad=array();
   $kaubad["vorst"]="2.50";
   $kaubad["juust"]="3.00";
   $kaubad["kartul"]="0.45";
?>
<!doctype html>
<html>
  <head>
     <title>Arvutamine</title>
  </head>
  <body>
     <h1>Summa kalkulaator</h1>
     <form action="?">
      Kaup:
       <select name="hind">
            <option value="">Vali kaup ...</option>
        <?php
          foreach($kaubad as $nimetus => $hind){
                echo "<option value='$hind'>$nimetus</option>";
          }
        ?>
       </select>
          Ostetav kogus:
               <select name="kogus">
                  <option>Vali kogus</option>
                  <option>1</option>
                  <option>2</option>
                  <option>3</option>
                  <option>4</option>
                  <option>5</option>
               </select>
               <input type="submit" value="OK"  />
      </form>
      <?php
         if(empty($_REQUEST["hind"]) or empty($_REQUEST["kogus"])){
           echo "Ootan sisestust.";
         } else {
           echo $_REQUEST["hind"]*$_REQUEST["kogus"];
         }
      ?>
  </body>
</html>
```



Ülesandeid

- Pane näide tööle, muuda andmeid, kontrolli tulemust.
- Pane lehele kaks valikupaari kaupade ja koguste tarbeks. Näita kummagi paari tarbeks summa eraldi ning lõppu kogusumma.
- Koosta massiiv, mille võtmeteks on liini bussipeatuste kaugused algpeatusest ning väärtusteks vastavate bussipeatuste nimed. Koosta samade andmetega kaks eri nimedega rippmenüüd. Kasutaja valib kaks peatust, talle teatatakse, kui suur on nende peatuste vahe kilomeetrites.
- Lisaks eelmisele on kolmandas rippmenüüs valik, kas tegemist on tava-, kiir- või ekspressliiniga. Peidetud väärtustena kuuluvad sinna juurde kilomeetrihinnad. Arvestatakse kokku sõidu maksumus.

Mobiilimalli järgi veebilehestik

Nähtud käskude järgi veebilehe sisu kokkupanekuga saab üsna varsti hakkama. Põhjalikum kujundamine aga nõuab sageli suuremat süvenemist, et taustade värvid ja piltide laiused ilusti paika saaks. Kui tahta "oma ja head" kujundust saada, siis tuleb see tee ikka läbi käia. Vahel aga on mugav end võõraste sulgedega ehtida ning kasutada mõnd juba olemasolevat kujunduspõhja. Märksõna "HTML template" või "CSS template" alt võib neid veebist hulgem leida ning mõndagi neist lubatakse vabalt kasutada.

Üksiku teate lugemine

Veebilehestikus on sisu ikka mitme lehe jagu. Kujundus aga võiks kokkukuuluvatel lehtedel ühesugune või vähemasti sarnane olla. Enne suurema lehtedeploki ühendamist vaatame näidet kahe failiga. Ühes neist lihtsalt paljas teade, mis tahetakse vajalikult lehele kuvada. Teine mõninga kujundusega leht, kuhu siis teade sobivasse kohta sisse loetakse.

Teatefailis teade spordipäeva kohta:

`teade.txt`

Spordipäeva tõttu sel esmaspäeval tavalisi tunde ei toimu.

Kõik õpilased kohtuvad hommikul kell 9 palliplatsil.

Sisse lugeval lehel tavalise veebilehestiku andmed, lõpus lihtsalt `require`-käsklus soovitud teate

sisse võtmiseks.

Fail:

sisselugemine.php

```
<!doctype html>
<html>
  <head>
     <title>Tunniplaani leht</title>
  </head>
  <body>
     <h1>Esmaspäev</h1>
     <ol>
         <li>Matemaatika</li>
         <li>Ajalugu</li>
         <li>Laulmine</li>
       </ol>
       <?php
          require("teade.txt");
       ?>
  </body>
</html>
```

Leht paistab lehitsejas välja järgmiselt.

Esmaspäev

1. Matemaatika
2. Ajalugu
3. Laulmine

Spordipäeva tõttu sel esmaspäeval
tavalisi tunde ei toimu. Kõik õpilased
kohtuvad hommikul kell 9 palliplatsil.

Nii on mugav vajalikku teadet lihtsalt tekstifaili kirjutada. Ning teate näitamiseks mõeldud lehed teavad, kust seda näha saab.

Valmis kujundusmall mobiilile

Kui omal kujundamise soont ei ole või lihtsalt tahtmine viisaka välimusega lehestik kiiresti üles saada, siis tasub ette võtta mõni valmis kujundusmall. Näitena võeti Gringo-nimeline mobiililehestiku põhi.

http://mobifreaks.com/free-mobile-website-templates/gringo-mobi-free-mobile-website-template/

Tutvumiseks saab lehestiku lahti pakkida ning tema näitfaili töötamist imetleda.

Oma tunniplaanirakenduse tarbeks teeme avalehest koopia ning kujundame sellest esmaspäeva tundide leheküljе, paigutades ülaserva viited ka teiste päevade jaoks mõeldud failidele.

blankett.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta name="viewport" content="width=device-width; initial-scale=1.0;
maximum-scale=1.0;">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Tunniplaan</title>
    <link href="kujundus.css" rel="stylesheet" type="text/css" />
</head>

<body>
      <div id="header">
        <div class="nav">
            <ul>
                <li><a href="esmaspaev.php">E</a></li>
                <li><a href="teisipaev.php">T</a></li>
                <li><a href="kolmapaev.php">K</a></li>
```

```
                <li><a href="neljapaev.php">N</a></li>
                <li><a href="reede.php">R</a></li>
            </ul>
        </div>
    </div>
    <div class="clear"></div>
    <h2>Esmaspäev</h2>
    <p>
      <ol>
          <li>Matemaatika</li>
          <li>Ajalugu</li>
          <li>Laulmine</li>
        </ol>
      </p>
        <div class="nav2">
                    <p>Design by <a
href="http://www.mobifreaks.com">Mobifreaks.com</a></p>
        </div>
</body>
</html>
```

Kujundusfailist jätame alles vaid lõigud, mis on vajalikud ühe päeva blanketti alles jäänud elementide tarbeks.

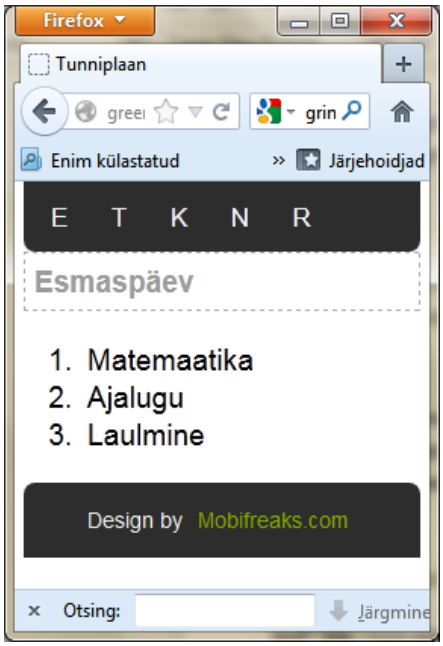
kujundus.css

```
@charset "utf-8";
/* CSS Document */
body{
      background:#ffffff;
      font-family:Arial, Helvetica, sans-serif;
      margin:0;
      padding:0;
      }
#header{
      margin:0 auto;
      }
.nav{
      font-size:14px;
    background:#2d2d2d;
    -moz-border-bottom-left-radius:6px;
    -webkit-border-bottom-left-radius:6px;
    border-bottom-left-radius:6px;
    -moz-border-bottom-right-radius:6px;
    -webkit-border-bottom-right-radius:6px;
    border-bottom-right-radius:6px;
    margin:0 5px;
      }
.nav ul{
      list-style-type:none;
    margin:0;
      padding:0;
      }
.nav ul li{
      display:inline;
      margin:0;
      padding:0;
      }
.nav ul li:first-child{
```

```
      margin:0 0 0 5px;
      }
.nav ul li:last-child{
      margin:0 5px 0 0;
      }
.nav ul li a{
      display:inline-block;
      color:#f2f2f2;
      padding:10px;
      text-decoration:none;
      }
.nav ul li a:hover{
      color:#565656;
      }

h2{
      font-size:16px;
      font-weight:bold;
      color:#9b9b9b;
      border:#aaaaaa 1px dashed;
      padding:5px;
      margin:0 5px;
      }
p{
      text-align:left;
      font-size:12px;
      color:#2d2d2d;
      margin:5px;
      padding:0;
      line-height:20px;
      }
p a{
    color:#3c9ddb;
    text-decoration:none;
      }
p a:hover{
    color:#4d4444;
    }

.nav2{
      font-size:14px;
    background:#2d2d2d;
    -moz-border-top-left-radius:6px;
    -webkit-border-top-left-radius:6px;
    border-top-left-radius:6px;
    -moz-border-top-right-radius:6px;
    -webkit-border-top-right-radius:6px;
    border-top-right-radius:6px;
    margin:0 5px;
      }
.nav2 p{
      text-align:center;
      color:#f2f2f2;
      margin:0 5px;
      padding:5px 0;
      }
.nav2 p a{
      display:inline-block;
      color:#88aa00;
      padding:5px;
      text-decoration:none;
      }
.nav2 p a:hover{
      color:#565656;
```

```
      }

.clear{
      clear:both;
      }
```

Need omavahel ühendatuna kuvavad välja ühe päeva jaoks kujundatud leheküljе.



Lehstiku juures soovime, et lehed oleksid sarnase kujundusega, kuid iga päeva leht eraldi sellele päevale vastava sisuga. Korduvad osad eraldame päisesse ja jalusesse, igale päevale vastava osa selle nädalapäeva nimelisse faili. Kõigepealt päisefail. Sinna jääb siis dokumendi algus, päiseosa ning menüüviited.

p2is.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta name="viewport" content="width=device-width; initial-scale=1.0;
maximum-scale=1.0;">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Tunniplaan</title>
    <link href="kujundus.css" rel="stylesheet" type="text/css" />
</head>

<body>
      <div id="header">
        <div class="nav">
            <ul>
                <li><a href="esmaspaev.php">E</a></li>
                <li><a href="teisipaev.php">T</a></li>
                <li><a href="kolmapaev.php">K</a></li>
                <li><a href="neljapaev.php">N</a></li>
                <li><a href="reede.php">R</a></li>
```

```
            </ul>
        </div>
    </div>
    <div class="clear"></div>
```

Jalusesse jääb muu hulgas viisakuse pärast viide lehemalli autorile.

jalus.php

```
<div class="nav2">
                  <p>Design by <a
href="http://www.mobifreaks.com">Mobifreaks.com</a></p>
        </div>
</body>
</html>
```

Edasi juba andmelehed päevade kaupa. Ülalt loetakse sisse päisefail, alt jalusefail.

esmaspaev.php

```
<?php require("p2is.php"); ?>
    <h2>Esmaspäev</h2>
    <p>
      <ol>
          <li>Matemaatika</li>
          <li>Ajalugu</li>
          <li>Laulmine</li>
        </ol>
      </p>
<?php require("jalus.php"); ?>
```

teisipaev.php

```
<?php require("p2is.php"); ?>
    <h2>Teisipäev</h2>
    <p>
      <ol>
          <li>Emakeel</li>
          <li>Matemaatika</li>
          <li>Tööõpetus</li>
        </ol>
      </p>
<?php require("jalus.php"); ?>
```

kolmapaev.php

```
<?php require("p2is.php"); ?>
    <h2>Kolmapäev</h2>
```

```
    <p>
      <ol>
          <li>Matemaatika</li>
          <li>Emakeel</li>
          <li>Kehaline kasvatus</li>
        </ol>
      </p>
<?php require("jalus.php"); ?>
```

neljapaev.php

```
<?php require("p2is.php"); ?>
    <h2>Neljapäev</h2>
    <p>
      <ol>
          <li>Emakeel</li>
          <li>Ajalugu</li>
          <li>Matemaatika</li>
        </ol>
      </p>
<?php require("jalus.php"); ?>
```

reede.php

```
<?php require("p2is.php"); ?>
    <h2>Reede</h2>
    <p>
      <ol>
          <li>Laulmine</li>
          <li>Ajalugu</li>
        </ol>
      </p>
<?php require("jalus.php"); ?>
```

Tulemusena saab liigelda päevade lehtede vahel ning ühtsena kujundatud lehestikku kasutada.

Ülesandeid

- Koosta eraldi tekstifail teate jaoks ning tekstifail lehe tegija nimega. Loo uus veebileht, kus ülal on pealkiri "Värske teade", lehe keskel failist sisseloetud teade ise ning all paremal lehe

tegija sisseloetud andmed.

- Otsi veebist mõni valmiskujundusega lehemall. Pane oma masinasse/serverisse tööle. Kujunda leht ühe anekdoodi näitamiseks. Pane lehe külge menüü viitega mitmele (veel olematule) anektoodilehele. Jaga eraldi failidesse päiseosa, sisu ja jalus. Loo mitme nalja jaoks eraldi sisu. Kujunda kokku ühiseks lehestikuks.

Andmebaasitabeli veebiväljund

Kord valmis tehtud veebilehti saab mugavasti veebist pärast vaadata. Kui aga tahta lehtede sisu vahetevahel muuta, või siis vastavalt kasutaja soovidele mitmesuguses järjestuses või moel kuvada - sellisel puhul aitavad andmebaaside võimalused lehestiku loomisele märgatavalt kaasa.

Enamikes andmebaasides paiknevad andmed relatsiooniliselt ehk tabelite kujul. Lihtsamal juhul on veebirakenduse juures tegemist ühe andmetabeliga. PHPga koos kasutatakse sageli MySQLi nimelist andmebaasiprogrammi, sest nad on mõeldud suhteliselt sarnasele sihtgrupile. Kokku mõned gigabaidid andmeid salvestatuna ning mõned päringud sekundis on süsteemile üldiselt jõukohased. Mahtude kasvamisel kordades aga tasub juba põhjalikuma serveri optimeerimise või muude vahendite peale mõelda.

Harjutamiseks saab serveri mugavasti püsti WAMP või XAMPP-nimelise komplekti abil. Andmebaasi kasutajaliidese eest aitab sealjuures hoolitseda PhpMyAdmin. Samas enamasti võimalik suhtlus ka käsurealt.

SQL

Andmebaasiga suhtlemiseks kasutatakse SQL-keelt. Selles leiduvad käsud andmetabelite loomiseks, sinna andmete lisamiseks, andmete küsimiseks, muutmiseks ja kustutamiseks.

Tabeli loomiseks käsklus CREATE TABLE. Käsu nimele järgneb tabeli nimi (praegusel juhul `lehed`). Ning siis sulgudes komadega eraldatuna tulpade nimed ning nende parameetrid. Iga tabeli esimeseks tulbaks on üldjuhul `id` - identifikaator, mille abil hiljem ridu eristada ja neile viidata. Parameetrid võivad lihtsamate rakenduste puhul enamasti samaks jääda. Selgitused:

`INT` - täisarv
`NOT NULL` - väärtus ei tohi puududa
`AUTO_INCREMENT` - server arvutab lisamisel ise juurde sobiva seni veel kasutamata väärtuse
`PRIMARY KEY` - selle tulba väärtust kasutatakse edaspidi tabeli vastavale reale viitamisel (näiteks muutmise või kustutamise juures).

Tulp `pealkiri` siin näites tüübiga `VARCHAR(50)` ehk siis tekst pikkusega kuni 50 tähte. Sisu tüübiks `TEXT`, mis tähendab, et pikkust ei piirata.

Kokku siis lause järgmine, mis tasub valmis kirjutada ning MySQLi käsuviibale või PHPMyAdmini aknasse kopeerida:

```
CREATE TABLE lehed(
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  pealkiri VARCHAR(50),
  sisu TEXT
);
```

Kui vastuseks tuli Query OK, siis järelikult ettevõtmine õnnestus. Muidu tuleb veateateid uurida ja uuesti proovida.

Andmete lisamiseks on loodud käsklus `INSERT INTO`. Järgneb tabeli nimi, siis sulgudes tulpade nimed, kuhu lisatavad andmed tulevad. Edasi sõna `VALUES` ning sulgude sisse komadega eraldatult igale tulbale vastav väärtus. Tekstilised andmed paigutatakse ülakomade vahele.

```
INSERT INTO lehed (pealkiri, sisu) VALUES ('Ilmateade', 'Kuiv ilm');
```

Tahtes rohkem andmeid lisada, tuleb INSERT lauset lihtsalt mitu korda käivitada, igal korral eraldi andmed sisse pannes.

```
mysql> INSERT INTO lehed (pealkiri, sisu) VALUES ('Korvpall', 'Treening reedel
kell 18');
Query OK, 1 row affected (0.00 sec)
```

Kui mõned sees, siis on hea vaadata ja kontrollida, et mis sinna täpsemalt sai. Andmete küsimiseks on SQLis loodud käsklus SELECT. Tärn tähendab, et kuvatakse kõikide olemasolevate tulpade andmed. Sõnale FROM järgneb tabeli nimi ning käsu lõppu käivitamiseks semikoolon. Lehtede tabeli sisu tuleb siis välja järgnevalt.

```
mysql> SELECT * FROM lehed;
+----+-----------+-------------------------+
| id | pealkiri  | sisu                    |
+----+-----------+-------------------------+
|  1 | Ilmateade | Kuiv ilm                |
|  2 | Korvpall  | Treening reedel kell 18 |
+----+-----------+-------------------------+
2 rows in set (0.00 sec)
```

Nagu näha, `id`-väärtused on automaatselt ise pandud, kuna vastaval tulbal on juures omadus `AUTO_INCREMENT`.

Tahtes andmeid veel juurde panna, tuleb taas käivitada `INSERT`-lause sobivate andmetega. Teksti sisestamisele vastab kõige korrasoleku puhul MySQL taas "Query OK", lisades sinna vahel ka mõjutatud ridade arvu ja kulunud aja - tähtis pigem suuremate andmestike korral.

```
mysql> INSERT INTO lehed (pealkiri, sisu) VALUES ('Matemaatika', 'Homme
tunnikontroll');
Query OK, 1 row affected (0.00 sec)
```

SQLi selecti abil saab andmeid kergesti sobivas järjekorras ja kujul välja küsida. Kui lause lõppu lisatakse `ORDER BY` koos vastava tulba nimega, siis tulevad andmed välja selle tulba järgi tähestiku järjekorda panduna (kui vastav tulp oli tekstitulp).

```
mysql> SELECT * FROM lehed ORDER BY sisu;
+----+-------------+-------------------------+
| id | pealkiri    | sisu                    |
+----+-------------+-------------------------+
|  3 | Matemaatika | Homme tunnikontroll     |
|  1 | Ilmateade   | Kuiv ilm                |
|  2 | Korvpall    | Treening reedel kell 18 |
+----+-------------+-------------------------+
3 rows in set (0.00 sec)
```

Saab küsida ka ainult ühe rea väärtusi:

```
mysql> SELECT pealkiri, sisu FROM lehed WHERE id=3;
+-------------+---------------------+
| pealkiri    | sisu                |
+-------------+---------------------+
| Matemaatika | Homme tunnikontroll |
+-------------+---------------------+
```

Kui leitakse, et rida pole enam vajalik, siis selle kustutamiseks sobib käsklus `DELETE`, kus soovitavalt `id` järgi määratakse ära, milline rida kustutada.

```
mysql> DELETE FROM lehed WHERE id=3;
Query OK, 1 row affected (0.00 sec)
```

Uue `SELECT`-päringuga saab kontrollida, mis siis sinna tegelikult alles jäi. Kui nüüd juhtutaks `INSERT`-lausega taas andmeid lisama, siis sellele reale enam id väärtust 3 välja ei antaks - välistamaks näiteks olukorda, kus vanale teatele pandud kommentaarid satuksid uue külge. Primaarvõtmetulba `id` väärtuseks tuleks uue rea lisamisel vähemasti 4.

```
mysql> SELECT * FROM lehed;
+----+-----------+-------------------------+
| id | pealkiri  | sisu                    |
+----+-----------+-------------------------+
|  1 | Ilmateade | Kuiv ilm                |
|  2 | Korvpall  | Treening reedel kell 18 |
+----+-----------+-------------------------+
```

Ülesandeid

- Tee näide läbi, lisa veel mõned read ja kustuta neid.
- Loo tabel kassid tulpadega id, kassinimi, toon
- Lisa paar kassi
- Väljasta kassid
- Väljasta kassid toonide järjekorras
- Kustuta üks kass

Tabeli sisu vaatamine

Andmebaasitabelis kannatab andmeid hoida ning SQL-käskude või mõne haldusliidese kaudu saab neid ka sinna lisada või sealt vaadata. Tavakasutaja aga eeldab, et ta näeb või sisestab veebilehitseja aknast just seda mis talle vaja ning ei taha ega jõua end koormata mitmesuguste tehniliste trikkidega. Veebirakenduse loojate üks tähtis ülesanne ongi andmed sobival kujul kasutajale ette näidata ning samuti veebilehtedel tehtud muutused pärast tabelites järgmiste kasutuskordade tarbeks talletada.

Sarnaselt kui ise SQL-käsklusi andmebaasi käsureale kirjutades õnnestub andmeid lisada ja küsida, saab ka PHP andmebaasiga SQL-käskude kaudu sidet pidada. Üheks võimaluseks käske veebilehe koodist andmebaasini vahendada on teek nimega MySQL Improved. Nii nagu käsitsi andmebaasiga suheldes peab teadma, kus masinas baas asub, millise kasutajanime ja parooliga sinna ligi pääseb ning millise nimega baasiga on tegemist - samad andmed vaja teada ka PHP poolt ühendust luues. Kui PHP ja MySQL asuvad samas masinas, siis sobib baasiserveri nimeks localhost. Siin näites

pruugin kasutajanimeks ja parooliks juku ning kala. Ja baasi nimeks siin jukubaas2. Eks oma lahendust luues tule siis need väärtused sisse kirjutada, mis parajasti pruukida on või teenusepakkujalt antakse. XAMPP vaikimisi seadete korral sobib näiteks serveriks "localhost", kasutajaks "root", parooliks tühi tekst "" ning katsetada saab baasis nimega "test". Avalikuks väljapanekuks pole selline komplekt küll viisakas, aga oma arvutis toimetamiseks käib küll.

Edasi tuleb andmete kättesaamiseks mitu sammu ette võtta. Mõne vahendiga saab veidi lihtsamalt, aga MySQL Improved teegi eeliseks on, et kui andmed viisakalt ette valmistada ning andmete SQL-käskudesse panekuks kasutada eelkompileeritud käsklusi (prepared statement), siis pole karta, et pahatahtlikke sisestuste abil veebilehtedelt saaks suuremat kurja teha. Muidu on aastaid olnud probleemiks, et kavalad veebilehtedel sisse kirjutatud laused võivad serveris käima minna ning pahandust tekitada. Lihtsamal juhul oma rakenduse andmeid kustutades või muutes, kuid keerukamatel juhtudel võivad löögi alla sattuda ka teiste rakenduste andmed või lausa välised serverid, kui kord sisse murtud masinat edasiste rünnakute alusena kasutatakse. Seetõttu siis siin matejalis andmete vahendajaks MySQL Improved tüüpi objekt, mida luuakse käsuga new mysqli ja antakse vajalikud ühendumisparameetrid kaasa.

Järgneva prepare-lausega palutakse $yhendus-nimelises muutuja kaudu kättesaadaval mysqli-objektil ette valmistada SQL-lause lehtede andmetabelist id, pealkirja ja sisu küsimiseks. Edasine bind_result määrab, kuhu muutujatesse saadud andmed pannakse. Andmebaasiga suhtlevad vahendid tehakse nõnda, et nad suudaksid toimida ka väga suurte andmekoguste korral ning ei loeks ilmaasjata suuremat kogust väärtusi mällu. Näiteks miljon rida on andmebaasis hoidmise jaoks täiesti kõlbulik kogus. Korraga mällu lugemisel võtab miljon kirjet aga hulga megabaite ning sealt hiljem midagi vajalikku kätte saada võib tülikas olla.

Edasi tulev execute() paneb käskluse baasis käima. Õnnetusena ei hoiatata, kui see lause unustatakse, aga lihtsalt andmeid ei saa kätte.

Vahepeal on mõningane osa HTML-i lehe kujunduse kuvamiseks. Baasis tulevate andmetega hakatakse tegelema siis, kui saabub tsükkel while($kask->fetch()). Iga fetch-käsklus tõstab päringu vastuste juurest ühe rea bind_param-käsuga määratud muutujatesse ning nendega võib tsüklikringi jooksul vajalikud toimetused ette võtta. Nõnda on korraga muutujate kaudu mälus vaid ühe andmerea ehk lehe andmed ning rakendus ei võta serveri mälu kuigivõrd. Praegu trükitakse pealkiri lihtsalt <h2> ja </h2> vahele ning näidatakse seetõttu suurema ja rasvasena välja. Sisu tuleb tavalise lõigu ehk div-ina. Käsk htmlspecialchars aitab hoolitseda, et kogemata andmete hulka sattunud erisümbolid (peamiselt < ja >) ei tekitaks lehe ülesehituse juures segadust.

Lehe väljastuse lõppemisel on viisakas andmebaasiühendus kinni panna.

Lehti väljastav kood tervikuna:

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas2");
  $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed");
  $kask->bind_result($id, $pealkiri, $sisu);
  $kask->execute();
?>
<!doctype html>
<html>
  <head>
     <title>Teated lehel</title>
  </head>
  <body>
     <h1>Teadete loetelu</h1>
     <?php
       while($kask->fetch()){
```

```
        echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
        echo "<div>".htmlspecialchars($sisu)."</div>";
      }
    ?>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Ning pilt valminud veebilehest:

Teadete loetelu

Ilmateade

Kuiv ilm

Korvpall

Treening reedel kell 18

Matemaatika

Homme tunnikontroll

Emakeel

Kontrolltöö

Ülesandeid

- Tee näide läbi
- Loo/otsi üles tabel kassid (id, kassinimi, toon). Näita kasside andmed veebilehele.
- Pane kasside toonideks inglisekeelsed värvinimetused
- Näita iga kass lehel vastavat värvi.

Teadete valik

Mõnekümne kassi andmed mahuvad ühele lehele ära. Aga seda vaid juhul, kui näidatakse vaid kassi nime ja värvi. Kui juba lisada omaniku andmed ning veidigi suurem kassi pilt, siis veidigi väiksema ekraani peal on mugav juba kasse ühekaupa vaadata. Järgnevalt uurimegi, kuidas selliseid lehti koostada, kus võimalik tabeli ühe rea andmeid eraldi välja tuua.

Üsna mugav on lehele andmeid saata aadressiriba kaudu. Kui kirjutan failinimele taha küsimärgi ning sinna taha id=2 ehk siis nt. teadetevalik.php?id=2 , siis selle väärtuse 2 saan programmis küsida muutujast $_REQUEST["id"]. Või kui tahan kontrollida, kas failinime järel saadeti

parameeter nimega id, siis kontrollin `if(isSet($_REQUEST["id"]))`

Nõnda ka järgmises lõigus. Kui parameeter saadeti, siis järelikult soovitakse vaadata ühe konkreetse lehe andmeid, mida id näitab. Kui aga parameetrit pole, siis inimene järelikult ei tea veel lehte selle numbri järele küsida ning tal on põhjust pigem loetelust omale sobiv valida.

Kui id on olemas, siis saab selle järgi küsida lehe muud andmed - praeguses näites pealkirja ja sisu. Tavalise SQL-lause juures saab ühe rea küsimiseks panna WHERE-tingimuse juurde vastava piirangu. Nt `SELECT id, pealkiri, sisu FROM lehed WHERE id=2;`

Kuna siin veebirakenduses tahetakse vastavalt kasutaja valikult näha erinevaid lehti, siis peab saama seda arvu muuta. MySQL Improved teek lubab muutuva väärtuse kohale panna küsimärgi ning pärast selle väärtuse bind_param-käsu abil asendada. Hiljem tulev rida

```
$kask->bind_param("i", $_REQUEST["id"]);
```

teatab, et parameetri tüübiks on täisarv ehk integer ehk täht i. Ning parameeter saab oma väärtuse muutujast `$_REQUEST["id"]`. Edasi juba andmete kättesaamine bind_result kaudu määratud muutujatesse nagu ennegi. Eelnevas näites võis andmeid tulla palju ning seetõttu tuli nad while-tsükli kaudu välja kuvada. Ühe id järgi küsides saab kätte ainult ühe rea, seetõttu piisab selle kättesaamiseks ühest fetch-käsklusest. Kas küsimine õnnestus, seda annab teada if-lause. Andmeid ei saa küsides näiteks juhul, kui keegi on aadressirea kaudu sisestanud olematu lehe id-numbri. Muul juhul saab pealkirja ja sisu ilusti kätte ning neid võib lehel kuvada.

```
if(isSet($_REQUEST["id"])){
   $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed
      WHERE id=?");
      //Kysim2rgi asemele pannakse aadressiribalt tulnud id,
      //eeldatakse, et ta on tyybist integer (i).
      //(double - d, string - s)
   $kask->bind_param("i", $_REQUEST["id"]);
   $kask->bind_result($id, $pealkiri, $sisu);
   $kask->execute();
   if($kask->fetch()){
     echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
     echo htmlspecialchars($sisu);
   } else {
     echo "Vigased andmed.";
   }
} else {
  echo "Tere tulemast avalehele! Vali men&uuml;&uuml;st sobiv teema.";
}
```

Tavakasutaja ei pea peast lehtede numbreid teadma. Tema pigem vaatab neid menüüst ning valib sobiva. Edasi juba saadetakse vastava teate id-number aadressiriba kaudu lehele, leht avaneb uuesti ning näitab küsitud teate sisu. Menüü kokku saamiseks sobib järgnev koodilõik. Kui viites (a href) jätta faili nime kohale küsimärk, siis avatakse sama fail ilma, et peaks selle faili nime teadma.

```
<?php
   $kask=$yhendus->prepare("SELECT id, pealkiri FROM lehed");
   $kask->bind_result($id, $pealkiri);
   $kask->execute();
   while($kask->fetch()){
     echo "<li><a href='?id=$id'>".
           htmlspecialchars($pealkiri)."</a></li>";
   }
?>
```

Edasi kirjete kaupa näitav kood tervikuna.

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas2");
?>
<!doctype html>
<html>
  <head>
    <title>Teated lehel</title>
    <style type="text/css">
       #menyykiht{
         float: left;
         padding-right: 30px;
       }
       #sisukiht{
         float:left;
       }
       #jalusekiht{
         clear: left;
       }
    </style>
      <meta charset="utf-8" />
  </head>
  <body>
    <div id="menyykiht">
        <h2>Teated</h2>
        <ul>
          <?php
             $kask=$yhendus->prepare("SELECT id, pealkiri FROM lehed");
             $kask->bind_result($id, $pealkiri);
             $kask->execute();
             while($kask->fetch()){
               echo "<li><a href='?id=$id'>".
                     htmlspecialchars($pealkiri)."</a></li>";
             }
          ?>
        </ul>
    </div>
    <div id="sisukiht">
       <?php
         if(isSet($_REQUEST["id"])){
            $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed
               WHERE id=?");
               //Kysim2rgi asemele pannakse aadressiribalt tulnud id,
               //eeldatakse, et ta on tyybist integer (i).
               //(double - d, string - s)
            $kask->bind_param("i", $_REQUEST["id"]);
            $kask->bind_result($id, $pealkiri, $sisu);
            $kask->execute();
            if($kask->fetch()){
              echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
              echo htmlspecialchars($sisu);
            } else {
              echo "Vigased andmed.";
            }
         } else {
           echo "Tere tulemast avalehele! Vali men&uuml;&uuml;st sobiv teema.";
         }
       ?>
    </div>
    <div id="jalusekiht">
       Lehe tegi Jaagup
    </div>
```

```
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Veebilehel kõigepealt näha menüü ning soovitus teema valida

Tere tulemast avalehele! Vali menüüst sobiv teema.

Teated

- Ilmateade
- Korvpall
- Matemaatika
- Emakeel

Lehe tegi Jaagup

Edasi juba tuleb vastava teema id aadressiribale ning näeb valitud pealkirja all peituvat sisu.

Teated

- Ilmateade
- Korvpall
- Matemaatika
- Emakeel

Korvpall

Treening reedel kell 18

Lehe tegi Jaagup

Ülesandeid

- Tee näide läbi
- Loo tabel koerte kohta (id, koeranimi, kirjeldus, pildiaadress)
- Sisesta andmed mõnede koerte kohta, pildid otsi veebist
- Loo lehestik, kus vasakus servas näha koerte nimed. Nimele vajutades kuvatakse lehel suurelt see koer koos pildi ja kirjeldusega.

Andmete lisamine ja kustutamine

Veebi kaudu on andmeid ilus vaadata. Ainult, et sellisena saab lehed ka ilma serveripoolse programmeerimistoeta tööle panna. Vajadusel saab kindla arvu lehti kopeerida ning viited vastavalt sättida ning võibki andmeid soovitult lugeda. Kui aga tahta, et kasutajapoolsed andmed ka kuidagi serverisse talletuks ning teised neid lugeda saaks - seda juba naljalt ilma serveripoolse programmita teha ei õnnestu. Muidugi kaasnevad serveris talletamisega ka omad mured: keegi võib hooletusest või pahatahtlikkusest sinna hulgem andmeid saata ning sellega serveris oleva andmebaasi täis kirjutada või lihtsalt kahtlaste postitustega suure hulga segadust tekitada. Aga hea ja halb käivad käsikäes ning mugavuse nimel tuleb vahel ka mõnevõrra riskida. Abilisteks hiljem varukoopiad, registreerimised, modereerimised ja muud täiendused.

Andmete lisamiseks tuleb need kõigepealt kasutajalt kätte saada. Selleks sobib sisestusvorm - olgu siis pidevalt lehel nähtaval või eraldi viite peale näidatav. Sisestusvormist tulevad andmed saadetakse salvestamiseks serverisse. Siinses näites toimib kõik sama faili kaudu, kuid iseenesest võib toimetuse jaoks ka eraldi teine väike fail loodud olla. Pärast andmete salvestamist on kasulik leht uuesti edasi suunata - kas või samale lehele, aga nõnda, et inimese sisestatud andmed uuesti kaasa ei tuleks - sellisel juhul pole karta, et värskendusnupu vajutamine andmeid korduvalt salvestama hakkab.

Lisamisvormi nähtavaks muutumiseks loodi viide parameetriga lisamine.

```
<a href='?lisamine=jah'>Lisa ...</a>
```

Kui selline parameeter jõuab serverisse, siis näidatakse kasutajale tühjad lahtrid, kuhu oma andmed kirja panna. Definition list (dl) koos nimetuse (definition term, dt) ning sisuga (dd, definition data) võimaldab mugavalt sisestuselemendid koos seletustega välja kuvada. Kaasas on ka varjatud element nimega `uusleht`, mille abil siis hiljem kontrollida, et kasutaja on uue lehe andmed saatnud. Andmete teele panekuks veebis nupp tüübist submit.

```
 if(isSet($_REQUEST["lisamine"])){
   ?>
     <form action='?'>
      <input type="hidden" name="uusleht" value="jah" />
      <h2>Uue teate lisamine</h2>
      <dl>
        <dt>Pealkiri:</dt>
        <dd>
         <input type="text" name="pealkiri" />
        </dd>
        <dt>Teate sisu:</dt>
        <dd>
          <textarea rows="20" name="sisu"></textarea>
        </dd>
       </dl>
       <input type="submit" value="sisesta">
     </form>
   <?php
 }
?>
```

Nupule vajutades avatakse leht uuesti. Kaasa liiguvad eelnevalt väljadesse sisestatud andmed. Eelnevalt varjatult kaasa pandud parameeter nimega uusleht näitab, et nüüd on paras aeg saabuvad andmed tabelisse kirjutada. Andmete lisamiseks tabelisse on INSERT-lause. Lisatavate väärtuste kohta tulevad algul küsimärgid, bind_param-käsu abil paigutatakse nende asemele tegelikud

väärtused. Tekst "ss" bind_param-käsu esimese parameetrina näitab, et mõlemad saabuvad väärtused on stringi ehk teksti tüüpi. Väärtusteks on siis pealkiri ja sisu, mis $_REQUEST-muutujast sisse loetakse. Vältimaks lehe korduslaadimisel uuesti salvestamist, tasub Location-päisekäsuga lehe avamine edasi suunata - kas või samale lehele ($_SERVER[PHP_SELF]). Viisakasti siis andmebaasiühendus ka sealjuures kinni ning lehe avamisele lõpp - exit();

```
  if(isSet($_REQUEST["uusleht"])){
    $kask=$yhendus->prepare("INSERT INTO lehed (pealkiri, sisu) VALUES (?, ?)");
    $kask->bind_param("ss", $_REQUEST["pealkiri"], $_REQUEST["sisu"]);
    $kask->execute();
    header("Location: $_SERVER[PHP_SELF]");
    $yhendus->close();
    exit();
  }
```

Näites lisati ka kustutamise moodus. Eraldi vaatamise lehel sai juurde kustutamise viide, kus aadressiga suunatakse samale lehele ning antakse kaasa `kustutusid`.

```
            echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
            echo htmlspecialchars($sisu);
            echo "<br /><a href='?kustutusid=$id'>kustuta</a>";
```

Lehe päises kustutusid saabumisel käivitatakse `DELETE`-lause koos etteantud kirje numbriga.

```
  if(isSet($_REQUEST["kustutusid"])){
    $kask=$yhendus->prepare("DELETE FROM lehed WHERE id=?");
    $kask->bind_param("i", $_REQUEST["kustutusid"]);
    $kask->execute();
  }
```

Lisamis- ja kustutusvõimeline kood tervikuna.

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas2");
  if(isSet($_REQUEST["uusleht"])){
    $kask=$yhendus->prepare("INSERT INTO lehed (pealkiri, sisu) VALUES (?, ?)");
    $kask->bind_param("ss", $_REQUEST["pealkiri"], $_REQUEST["sisu"]);
    $kask->execute();
    header("Location: $_SERVER[PHP_SELF]");
    $yhendus->close();
    exit();
  }
  if(isSet($_REQUEST["kustutusid"])){
    $kask=$yhendus->prepare("DELETE FROM lehed WHERE id=?");
    $kask->bind_param("i", $_REQUEST["kustutusid"]);
    $kask->execute();
  }
?>
<!doctype html>
<html>
  <head>
    <title>Teated lehel</title>
    <style type="text/css">
       #menyykiht{
         float: left;
         padding-right: 30px;
       }
       #sisukiht{
         float:left;
```

```
      }
      #jalusekiht{
        clear: left;
      }
    </style>
  </head>
  <body>
    <div id="menyykiht">
        <h2>Teated</h2>
        <ul>
          <?php
             $kask=$yhendus->prepare("SELECT id, pealkiri FROM lehed");
             $kask->bind_result($id, $pealkiri);
             $kask->execute();
             while($kask->fetch()){
               echo "<li><a href='?id=$id'>".
                           htmlspecialchars($pealkiri)."</a></li>";
             }
          ?>
        </ul>
        <a href='?lisamine=jah'>Lisa ...</a>
    </div>
    <div id="sisukiht">
       <?php
         if(isSet($_REQUEST["id"])){
            $kask=$yhendus->prepare("SELECT id, pealkiri, sisu FROM lehed
               WHERE id=?");
            $kask->bind_param("i", $_REQUEST["id"]);
            $kask->bind_result($id, $pealkiri, $sisu);
            $kask->execute();
            if($kask->fetch()){
              echo "<h2>".htmlspecialchars($pealkiri)."</h2>";
              echo htmlspecialchars($sisu);
              echo "<br /><a href='?kustutusid=$id'>kustuta</a>";
            } else {
              echo "Vigased andmed.";
            }
         }
         if(isSet($_REQUEST["lisamine"])){
           ?>
             <form action='?'>
              <input type="hidden" name="uusleht" value="jah" />
              <h2>Uue teate lisamine</h2>
              <dl>
                <dt>Pealkiri:</dt>
                <dd>
                 <input type="text" name="pealkiri" />
                </dd>
                <dt>Teate sisu:</dt>
                <dd>
                  <textarea rows="20" name="sisu"></textarea>
                </dd>
               </dl>
               <input type="submit" value="sisesta">
             </form>
           <?php
         }
       ?>
    </div>
    <div id="jalusekiht">
       Lehe tegi Jaagup
    </div>
  </body>
</html>
```

```
<?php
  $yhendus->close();
?>
```

Lehel algul näha teadete loetelu.

Teated

- Ilmateade
- Korvpall
- Matemaatika
- Emakeel

Lisa ...
Lehe tegi Jaagup

Pealkirjale vajutades näeb vastava kirje andmeid. Kustutusviite kaudu saab kirjest lahti.

Teated

- Ilmateade
- Korvpall
- Matemaatika
- Emakeel

Matemaatika

Homme tunnikontroll
kustuta

Lisa ...
Lehe tegi Jaagup

Nii pole seda teadet ka enam menüüs näha. Paistab lisamisviide uute andmete sisestamiseks.

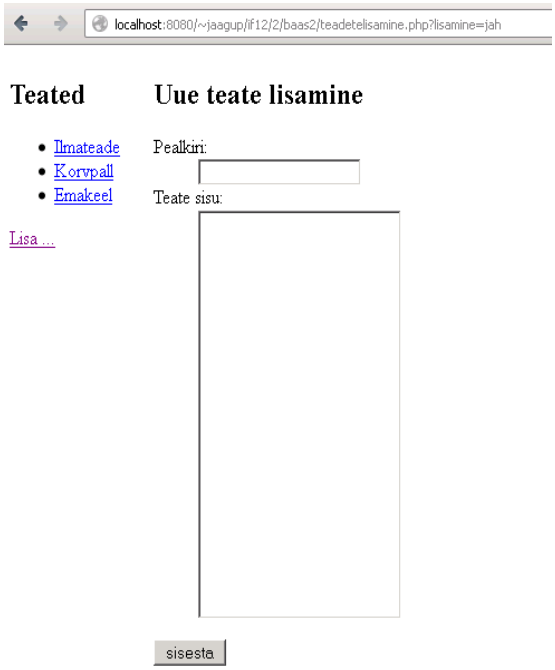
Teated

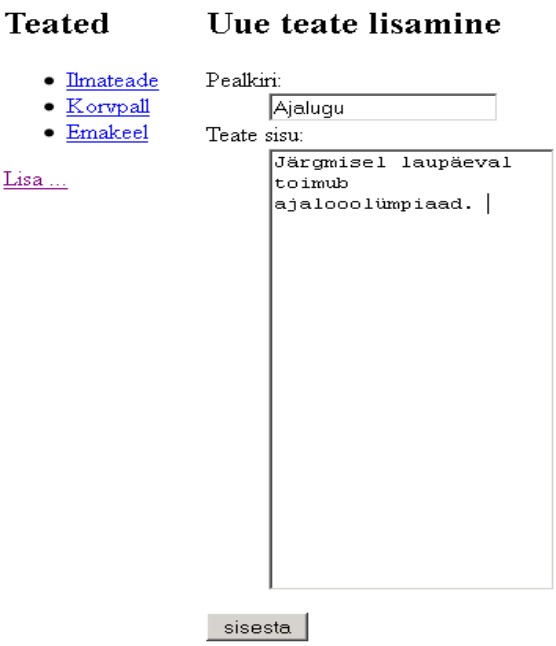
- Ilmateade
- Korvpall
- Emakeel

Lisa ...
Lehe tegi Jaagup

Lisamisviite kaudu kuvatakse lisamisvorm

Edasi tuleb need andmed sisse kirjutada.





Pärast sisestamist võibki menüüs uud teadet imetleda.

Teated

- Ilmateade
- Korvpall
- Ajalugu
- Emakeel

Lisa ...
Lehe tegi Jaagup

Ülesandeid

- Tee näide läbi
- Loo/otsi koerte tabel (id, koeranimi, kirjeldus, pildiaadress)
- Võimalda koeri veebi kaudu lisada, vaadata ja kustutada

Peoõhtu registreerimisvorm

- Koosta veebileht peokuulutusega
- Loo sinna juurde veebileht, kus kasutaja saab oma eesnime, perekonnanime ja elektronposti sisestada. Andmed talletatakse tabelisse (ei näidata veebilehel).
- Loo eraldi administraatorileht, kus saab sisestusi näha (sisselogimist pole vaja)
- Administraator saab vigaseid sisestusi ka kustutada
- Loo teine andmetabel, kus kirjas peo etteasted ja sündmused koos arvatava kellaajaga.

Väljasta andmed kellaaegade järjekorras eraldi veebilehele. Kujunda veebileht koos eelmistega ühtseks lehekstikuks.

- Loo eraldi administraatorileht peo sündmuste lisamiseks ja kustutamiseks.

Andmete muutmine, laulude lehestik

Lisamise ja kustutamisega saab küllalt palju ühekordseid registreerimisi ja vahetamisi korda ajada. Põhjalikumate rakenduste juures aga käivad sama andmerea eri tulbad eri kasutajate juurest läbi ning siis tuleb ka rea andmete muutmisele mõelda nii, et osa väärtusi säilib, mõned uuenevad. Siin näitena koostame laulude lehestiku, kus haldur saab neid tabelisse lisada, kasutajad lauludele punkte ja kommentaare jagada ning halduril pärast võimalik määrata, millised laulud parajasti välja paistavad ja millised mitte. Ning kõik see lehestik ehitatakse ühe andmetabeli peale, kus siis eri veerugude väärtusi saab sobivalt kasutada.

Sellise lehestiku baasipooleks piisab, kui laulude andmed on andmebaasis, tabelis nimega

laulud(id, pealkiri, punktid, lisamisaeg, kommentaarid, avalik)

SQL-lause tabeli loomiseks:

```
CREATE TABLE laulud(
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  pealkiri VARCHAR(50),
  punktid INT DEFAULT 0,
  lisamisaeg DATETIME,
  kommentaarid TEXT,
  avalik INT DEFAULT 1
);
```

Sõna DEFAULT tulba taga määrab vaikimisi väärtuse - ehk kui laul luuakse, siis pole tal veel ühtki punkti, aga samas on avalik. Lisamisaja andmetüüp DATETIME näitab, et üheaegselt hoitakse meeles kuupäev ja kellaaeg.

Lehestiku koostamise saab jagada suurematekseks alamülesanneteks:

* Looge veebileht laulude lisamiseks andmebaasi.

  Sisestada on vaja vaid pealkiri, tulemust näeb vaid baasist

* Looge veebileht lauludele plusspunkti andmiseks.

Iga laulu taga näeb talle antud punkte.

* Looge veebileht, mis näitaks vaid avalikke laule

Selle juures aitab SQL-i poolest päring laulu nr 1 näitel

SELECT pealkiri FROM laulud WHERE avalik=1

* Muutke andmebaasi käsklusega mõni laul peidetuks

Tarvilik SQL-käsklus sealjuures

UPDATE laulud SET avalik=0 WHERE id=1

Veenduge, et seda laulu avalike laulude lehele ei nähe

* Koosta haldusleht, mille abil on võimalik laule peita ja taas avalikuks muuta.

Uue laulu lisamine

Lisamiseks on vaja lisamisvormi ja salvestuskohta. Kui ülalpoolses näites kippus mõnevõrra segadust tekitama, et lisamisvorm avanes vaid vastaval viitel vajutades, siis siin püütud lisamine võimalikult lihtsaks teha. Lisamislahter on kohe lehe avamisel olemas. Ning samal lehel püütakse pealkiri kinni ja lisatakse andmed tabelisse. Lisamisaja väärtuseks NOW() annab serverikella praeguse aja. Kommentaariks lisatakse algul tühi tekst, pärast saab sinna kasutajate juttu juurde paigutada.

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas1");
  if(!empty($_REQUEST["uuepealkiri"])){
     $kask=$yhendus->prepare(
        "INSERT INTO laulud(pealkiri, lisamisaeg, kommentaarid) VALUES(?, NOW(),
' ')");
       $kask->bind_param("s", $_REQUEST["uuepealkiri"]);
       $kask->execute();
       echo $yhendus->error;
       header("Location: $_SERVER[PHP_SELF]");
       $yhendus->close();
       exit();
  }
?>
<!doctype html>
<html>
  <head>
    <title>Laulud</title>
  </head>
  <body>
    <h1>Laulud</h1>
      <form action="?">
         Uue laulu pealkiri:
           <input type="text" name="uuepealkiri" />
             <input type="submit" value="Lisa laul" />
```

```
      </form>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Kõigepealt ilmub pealkirja lisamise lahter

Sinna võib kirjutada uue laulu pealkirja, vajutada lisamisnuppu ning ongi see laul tabelis kirjas.

Ülesandeid

- Pane näide tööle.
- Koosta tabel koolipeole kutsutava ansambli valimiseks ja hääletamiseks.
  Tabel ansamblid(id, ansamblinimi, punktid, kommentaarid, avalik, otsus). Kommentaarid tüübist TEXT, otsus VARCHAR(255).
- Koosta leht ansambli nime lisamiseks tabelisse. Kontrolli lehe tööd.

Lauludele punktide lisamine

Punktide lisamis leht koosneb kahest osast. Kõigepealt kuvatakse loetelu kõigis tabelis olevatest lauludest. Ning kui kasutaja ühele neist vajutab, siis lisatakse laulule punkt. Punkti lisamiseks on viide

Lisa punkt

ehk siis küsimärgiga viide viib kasutaja samale failile. Kaasa antakse parameeter nimega healaulu_id, väärtuseks selle laulu id, mille nimele parajasti vajutatakse.

Serveris lehe uuel avamisel kontrollitakse, kas parameeter healaulu_id on olemas. Kui jah, siis pannakse sellele laulule üks punkt juurde, ehk suurendatakse vastava välja väärtust.

Väärtuse suurendamiseks ühe võrra sobib SQL-lause

```
UPDATE laulud SET punktid=punktid+1 WHERE id=?
```

kus siis küsimärgi kohale tuleb pärast käsu ettevalmistust vastava laulu id.

Punktide jagamise kood tervikuna

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas1");
  if(isSet($_REQUEST["healaulu_id"])){
     $kask=$yhendus->prepare("UPDATE laulud SET punktid=punktid+1 WHERE id=?");
       $kask->bind_param("i", $_REQUEST["healaulu_id"]);
       $kask->execute();
  }
?>
<!doctype html>
<html>
  <head>
    <title>Laulud</title>
  </head>
  <body>
    <h1>Laulud</h1>
    <table>
        <?php
           $kask=$yhendus->prepare("SELECT id, pealkiri, punktid FROM laulud");
             $kask->bind_result($id, $pealkiri, $punktid);
             $kask->execute();
             while($kask->fetch()){
                $pealkiri=htmlspecialchars($pealkiri);
                echo "<tr>
                    <td>$pealkiri</td>
                    <td>$punktid</td>
                    <td><a href='?healaulu_id=$id'>Lisa punkt</a></td>
                  </tr>";
             }
        ?>
      </table>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Tulemusena ilmuvad olemasolevad laulud silma ette.

Laulud

Mutionu	9	Lisa punkt
Mutionu pidu	8	Lisa punkt
Muumioru lood	4	Lisa punkt
Rongisõit	0	Lisa punkt

Millisele reale vajutati, selle laulu punktide arv suureneb ühe võrra.

Laulud

Mutionu	9	Lisa punkt
Mutionu pidu	8	Lisa punkt
Muumioru lood	4	Lisa punkt
Rongisõit	1	Lisa punkt

Ülesandeid

- Tee näide läbi
- Loo/otsi eelneva ülesande ansamblite tabel.
- Loo võimalus ka ansamblitele punktide jagamiseks.
- Tekita juurde tulbad, kus on võimalik punkte ka kahe- ja kolmekaupa jagada.

Laulude peitmine ja avalikustamine

"Päris" rakenduste juures enamasti ei kustutata midagi lihtsalt ära, vaid lihtsalt märgitakse, et vastav kirje/rida on "arhiveeritud". Selliselt on võimalik ka veebiandmete kaudu toimunu ajalugu vaadata ning vajadusel mõningaid kohti siluda/taastada. Kui millalgi on vaja surnud ridade arvel andmebaasi mahtu vähendada, siis need enamasti korjatakse kusagile varundustabelisse või faili kokku ning lastakse seal veel mõnda aega olla, kuni loota võib, et sealt enam midagi tähtsamat vajalikuks ei osutu. Sarnase avalikustamise ja peitmise mängime läbi ka laulude puhul. Kas laul on avalik või mitte, seda saab lugeda vastavast tulbast: 0 - peidetud, 1- avalik. Lehel avalike laulude näitamiseks pannakse laulude näitamise juures päringule lihtsalt WHERE-tingimus avalik=1.

avalikudlaulud.php

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas1");
?>
<!doctype html>
<html>
  <head>
    <title>Laulud</title>
  </head>
  <body>
    <h1>Laulud</h1>
    <table>
        <?php
           $kask=$yhendus->prepare(
               "SELECT id, pealkiri, punktid FROM laulud WHERE avalik=1");
           $kask->bind_result($id, $pealkiri, $punktid);
           $kask->execute();
           while($kask->fetch()){
              $pealkiri=htmlspecialchars($pealkiri);
```

```
                echo "<tr>
                    <td>$pealkiri</td>
                    <td>$punktid</td>
                  </tr>";
            }
        ?>
      </table>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Laulud esiotsa ilusasti avalikult näha.

Laulud

Mutionu	9
Mutionu pidu	8
Muumioru lood	4
Rongisõit	1

Laulude peitmiseks saab nende nimed ette kuvada ning taga oleva viite kaudu siis loo peidetuks muuta. Jällegi tuleb kaasa anda loo id ning lehe ülaosas uuel laadimisel sellele vastavalt reageerida. Vastavalt saadetud peitmise_id-le käivitatakse SQL-lause

UPDATE laulud SET avalik=0 WHERE id=?

mille tulemusena vajutatud viitega laulu tulbale avalik antakse väärtuseks 0 ning laulu enam avalike laulude seas ei kuvata.

haldus3.php

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas1");
  if(isSet($_REQUEST["peitmise_id"])){
     $kask=$yhendus->prepare("UPDATE laulud SET avalik=0 WHERE id=?");
       $kask->bind_param("i", $_REQUEST["peitmise_id"]);
       $kask->execute();
  }
?>
<!doctype html>
<html>
  <head>
    <title>Laulud</title>
  </head>
  <body>
    <h1>Laulud</h1>
```

```
    <table>
        <?php
           $kask=$yhendus->prepare("SELECT id, pealkiri, avalik FROM laulud");
             $kask->bind_result($id, $pealkiri, $avalik);
             $kask->execute();
             while($kask->fetch()){
                $pealkiri=htmlspecialchars($pealkiri);
                echo "<tr>
                    <td>$pealkiri</td>
                    <td>$avalik</td>
                    <td><a href='?peitmise_id=$id'>Peida</a></td>
                  </tr>";
             }
        ?>
      </table>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Algul kõik lood avalikud

Laulud

Mutionu	1	Peida
Mutionu pidu	1	Peida
Muumioru lood	1	Peida
Rongisõit	1	Peida

Pärast vajutust läks Muumioru lugude avalik-tulp nulliks.

Laulud

Mutionu	1	Peida
Mutionu pidu	1	Peida
Muumioru lood	0	Peida
Rongisõit	1	Peida

Tulemusena seda lugu avalike laulude all ei kuvata.

Laulud

Mutionu	9
Mutionu pidu	8
Rongisõit	1

Lihtsamal juhul piirduvadki lehe oskused vaid peitmisega. Näiteks kui vaja roppe veebikommentaare varju panna, siis tavalisel halduril võib täiesti piisata peitmise-viitest. Erandkorras tagasipaneku võib kasvõi eraldi väikese lehena ehitada. Kui aga soov mõlemas suunas määramised samale lehele panna, siis ka see võimalik ning nii siinses näites ka tehakse. Üheks mooduseks oleks teha lehele eraldi tulp peitmise, eraldi avalikustamise tarbeks. Mõngase sättimise tulemusena pääseb aga ühe tulbaga - lihtsalt tuleb viiteid ja sõnu nõnda kohendada, et vajutuse peale olemasolevas seisus muutus tekiks.

Lehe päises on parameetrite kohta kaks valikut. Kui tuleb peitmise_id, siis vastav laul peidetakse. Kui tuleb avamise_id, siis selle id-ga rida muudetakse nähtavaks.

Õige teksti ja viite näitamiseks sobib lõik

```
$avamistekst="Ava";
$avamisparam="avamise_id";
$avamisseisund="Peidetud";
```

Ehk siis algul eeldatakse, et lugu pole avalik, parameetri nimeks saab avamise_id ning kasutajale nähtav sõna on "Peidetud". Alloleval real trükitakse muutujate väärtused nõnda ka lehele.

```
<td><a href='?$avamisparam=$id'>$avamistekst</a></td>
```

Kui aga päringust selgub, et laul siiski on avalik, siis pööratakse muutujate väärtused ümber ning väljatrüki tulemusena tekib oluord, kus vajutuse tulemusena pannakse lugu peitu.

```
if($avalik==1){
   $avamistekst="Peida";
   $avamisparam="peitmise_id";
   $avamisseisund="Avatud";
}
```

Nii ongi võimalik samal kohal korduvalt klõpsides laulu seisundit avalikust peidetuks ja tagasi muuta.

haldus4.php

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas1");
  if(isSet($_REQUEST["peitmise_id"])){
     $kask=$yhendus->prepare("UPDATE laulud SET avalik=0 WHERE id=?");
       $kask->bind_param("i", $_REQUEST["peitmise_id"]);
       $kask->execute();
  }
```

```
  if(isSet($_REQUEST["avamise_id"])){
     $kask=$yhendus->prepare("UPDATE laulud SET avalik=1 WHERE id=?");
       $kask->bind_param("i", $_REQUEST["avamise_id"]);
       $kask->execute();
  }
?>
<!doctype html>
<html>
  <head>
    <title>Laulud</title>
  </head>
  <body>
    <h1>Laulud</h1>
    <table>
        <?php
           $kask=$yhendus->prepare("SELECT id, pealkiri, avalik FROM laulud");
             $kask->bind_result($id, $pealkiri, $avalik);
             $kask->execute();
             while($kask->fetch()){
                $pealkiri=htmlspecialchars($pealkiri);
                  $avamistekst="Ava";
                  $avamisparam="avamise_id";
                  $avamisseisund="Peidetud";
                  if($avalik==1){
                     $avamistekst="Peida";
                     $avamisparam="peitmise_id";
                     $avamisseisund="Avatud";
                  }
                echo "<tr>
                    <td>$pealkiri</td>
                    <td>$avamisseisund</td>
                    <td><a href='?$avamisparam=$id'>$avamistekst</a></td>
                  </tr>";
             }
        ?>
      </table>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Alustuseks näha et eelnevalt peidetud Muumioru lood on endiselt peidus.

Avamisviite peale tuleb lugu avalikuks

Laulud

Mutionu	Avatud	Peida
Mutionu pidu	Avatud	Peida
Muumioru lood	Avatud	Peida
Rongisõit	Avatud	Peida

Samuti tekib ta siis laulude üldisesse loendisse

Laulud

Mutionu	9
Mutionu pidu	8
Muumioru lood	4
Rongisõit	1

Ülesandeid

- Tee näide läbi
- Lisa ansamblite lehele avalikustamise ja peitmise võimalus.
- Võimalda eraldi peita ja näidata korraga kõiki neid ansambleid, kel pole veel ühtegi punkti.

Kommenteerimine

Veebilehtedele kirjutatakse kommentaare ja täiendusi päris mitmel puhul. Ajalehtedes kommenteeritakse uudiseid, tehnikud märgivad tehtud töid, siin püüame kokku koguda laulude kohta tehtavad kommentaarid. Keerukamal juhul tasub teha kommentaaride jaoks eraldi andmetabel, siis on võimalik neid mugavalt näiteks kirjutaja või loomisaja järgi järjestada. Lihtsamal juhul aga piisab iga laulu kohta lihtsalt ühest kommentaariväljast vastavas tulbas, kuhu inimeste kirjutatud kommentaarid üksteise otsa lisatakse.

Lisamise puhul tuleb kuidagi kindlaks määrata, et millise laulu juurde vastav kommentaar

kirjutatakse. Siin näites tehakse iga laulu juures olevasse lahtrisse eraldi vorm. Sinna sisse pannakse varjatud väli laulu kohta, millele uus kommentaar kirjutatakse. Edasi lisatakse tekstiväli ning sisestusnupp. Korraga saadetakse veebilehitsejast serverisse vaid ühe vormi andmed - just selle omad, kus sisestusnuppu vajutati. Sellise trikiga saabki hoolitseda, et soovitud laulu id läheb koos kommentaaritekstiga kaasa.

```
  <td>
    <form action='?'>
         <input type='hidden' name='uue_kommentaari_id' value='$id' />
         <input type='text' name='uus_kommentaar' />
         <input type='submit' value='Lisa kommentaar' />
    </form>
  </td>
```

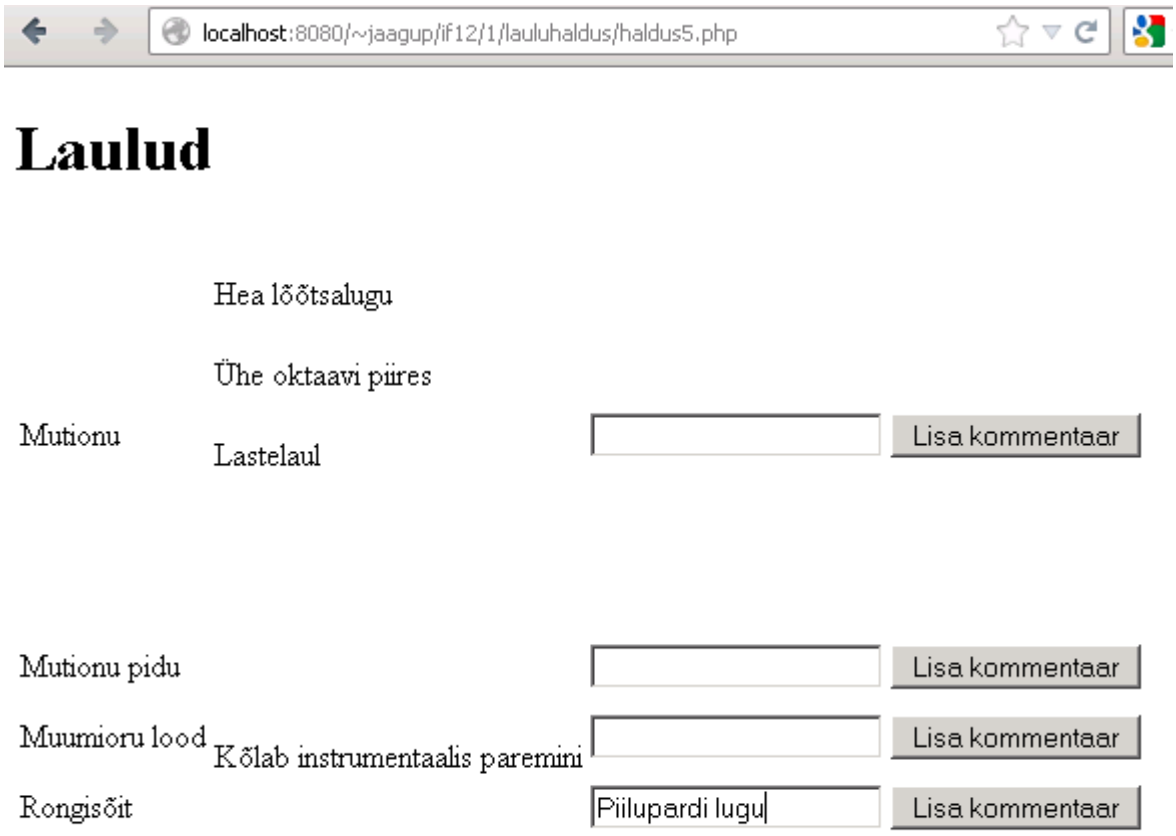
Lehe päises vaadatakse, kas saabus uue kommentaari id. Kui jah, siis lisatakse saabunud kommentaari teksti selle laulu kommentaarilahtri teksti lõppu ja pannakse reavahetus vahele. Lehe sisu avanemisel saab nõnda juba uut kommentaari näha. Käsk htmlspecialchars asendab erisümbolid, `nl2br` asendab tekstis olevad reavahetused HTML-i `<br />`-käskudega.

haldus5.php

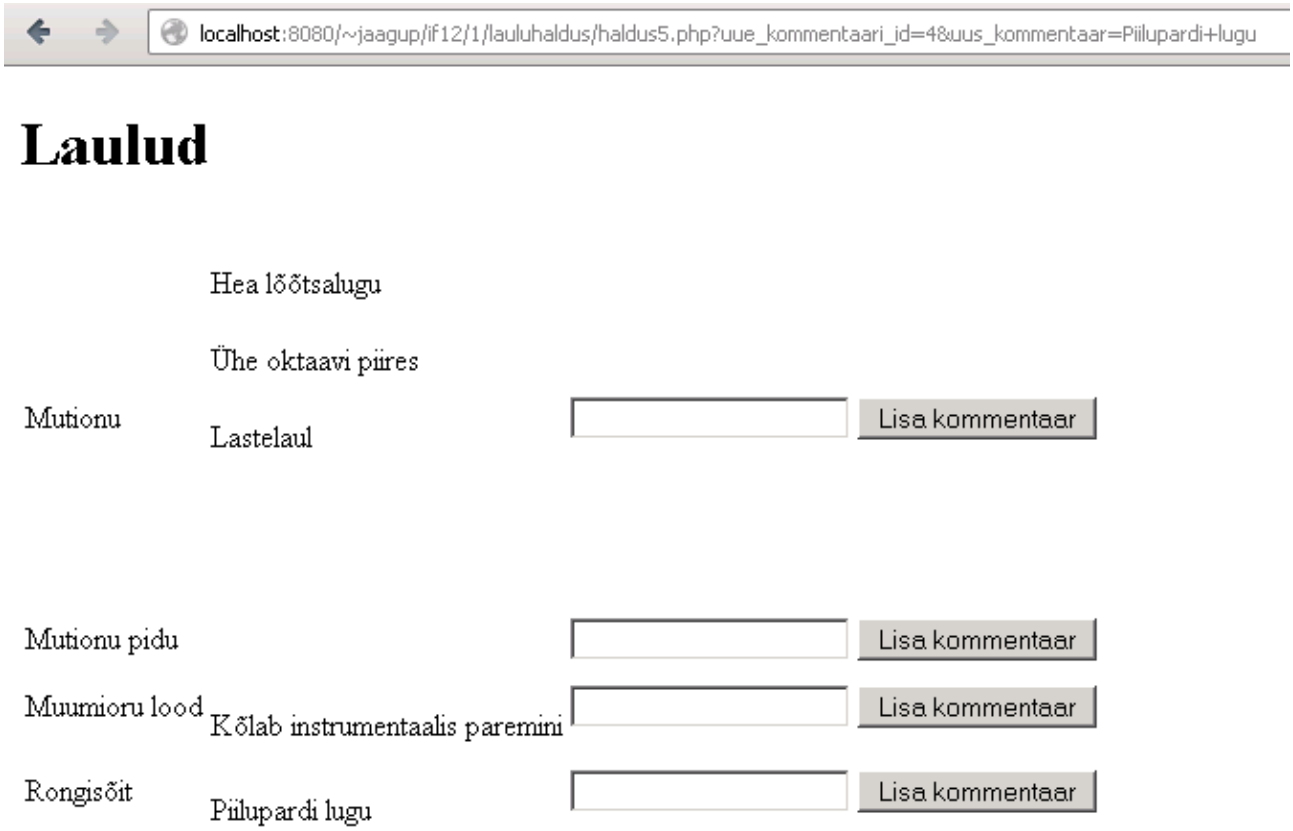
```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas1");
  if(isSet($_REQUEST["uue_kommentaari_id"])){
     $kask=$yhendus->prepare(
         "UPDATE laulud SET kommentaarid=CONCAT(kommentaarid, ?)  WHERE id=?");
       $kommentaarilisa="\n".$_REQUEST["uus_kommentaar"]."\n";
       $kask->bind_param("si", $kommentaarilisa,
$_REQUEST["uue_kommentaari_id"]);
       $kask->execute();
  }
?>
<!doctype html>
<html>
  <head>
    <title>Laulud</title>
  </head>
  <body>
    <h1>Laulud</h1>
    <table>
        <?php
           $kask=$yhendus->prepare(
               "SELECT id, pealkiri, kommentaarid FROM laulud");
             $kask->bind_result($id, $pealkiri, $kommentaarid);
             $kask->execute();
             while($kask->fetch()){
                $pealkiri=htmlspecialchars($pealkiri);
                  $kommentaarid=nl2br(htmlspecialchars($kommentaarid));
                echo "<tr>
                    <td>$pealkiri</td>
                    <td>$kommentaarid</td>
                    <td>
                      <form action='?'>
                           <input type='hidden' name='uue_kommentaari_id'
value='$id' />
                           <input type='text' name='uus_kommentaar' />
```

```
                        <input type='submit' value='Lisa kommentaar' />
                    </form>
                </td>
            </tr>";
        }
    ?>
  </table>
 </body>
</html>
<?php
 $yhendus->close();
?>
```

Tulemusena siis teikib iga laulu nime järele kommentaari lisamise lahter. Vahepealses tulbas näha laulule eelnevalt lisatud kommentaarid. Lahtrisse võib lisada uue kommentaari.



Pärast lisamisnupule vajutamist näeb seda kommentaari juba laulu nime järel.



Ülesandeid

- Tee näide läbi
- Lisa kommenteerimisvõimalus ansamblivaliku lehele
- Lisa kommentaarile automaatselt ka aeg PHP date-käskluse abil

Haldus laulude kaupa

Tähelepanekud õppijate juures kipuvad näitama, et kui ühe toiminguga lehest eraldi aru saadud ning mõistetakse selle põhjal ka sarnaseid lehti teisel teemal koostada - sellest veel ei pruugi piisata, et mitme toimetuse ühele lehele kokkupanek sama hõlpsasti käiks. Seetõttu ka siin näide, kuidas punktide haldus ning kommentaaride lisamine võimalik samale lehele kokku tõsta. Lihtsamal kujul käiks see nõnda, et laulu taga on kaks tulpa - üks punktide määramiseks ning teine kommentaari lisamiseks. Ning üleval oleks vastavad andmetabeli muutmise plokid järjestikku, if-lausega saab kontrollida, et kumma toiminguga parajasti tegemist.

Siin aga on keerukamalt ette võetud ning tehtud nõnda, et laulude loetelu on eraldi ning sealt valitud laulu andmed näidatakse lehe ülaosas. Kuna korraga aktiivne vaid üks laul, siis jagub lehel rohkem ruumi temaga seotud ettevõtmiste tarbeks.

Laulude loetellu küsitakse kasutaja lehele välja avalikud laulud. Iga laulu pealkiri muutub väljatrükil viiteks, mis näitab samale veebilehele, kuid kuhu antakse kaasa valitud laulu id (`href='?id=$id'`).

```
    <table>
```

```
        <?php
           $kask=$yhendus->prepare(
                "SELECT id, pealkiri, punktid FROM laulud WHERE avalik=1");
             $kask->bind_result($id, $pealkiri, $punktid);
             $kask->execute();
             while($kask->fetch()){
                $pealkiri=htmlspecialchars($pealkiri);
                echo "<tr>
                    <td><a href='?id=$id'>$pealkiri</a></td>
                    <td>$punktid</td>
                  </tr>";
             }
        ?>
      </table>
```

Kui laulu id valiti, sellisel juhul küsitakse lehel pärast body algust välja päring vaid selle laulu andmete kätte saamiseks.

```
        if(isSet($_REQUEST["id"])){
           $kask=$yhendus->prepare("SELECT id, pealkiri, kommentaarid, punktid,
lisamisaeg FROM laulud WHERE id=?");
```

Sealt edasi võib siis ka laulu kommentaari sisestada või punkti lisada.

Andmete tegelik baasi kirjutamine toimub lehe uuel avanemisel kui vastavad parameetrid kaasas. See osa näha kohe uue väljatrüki alguses.

laululeht.php

```
<?php
  $yhendus=new mysqli("localhost", "juku", "kala", "jukubaas1");
  if(isSet($_REQUEST["healaulu_id"])){
     $kask=$yhendus->prepare("UPDATE laulud SET punktid=punktid+1 WHERE id=?");
       $kask->bind_param("i", $_REQUEST["healaulu_id"]);
       $kask->execute();
  }
  if(isSet($_REQUEST["uue_kommentaari_id"])){
     $kask=$yhendus->prepare(
         "UPDATE laulud SET kommentaarid=CONCAT(kommentaarid, ?)  WHERE id=?");
       $kommentaarilisa="\n".$_REQUEST["uus_kommentaar"]."\n";
       $kask->bind_param("si",
               $kommentaarilisa, $_REQUEST["uue_kommentaari_id"]);
       $kask->execute();
  }
?>
<!doctype html>
<html>
  <head>
    <title>Laulud</title>
  </head>
  <body>
    <?php
        if(isSet($_REQUEST["id"])){
           $kask=$yhendus->prepare(
             "SELECT id, pealkiri, kommentaarid, punktid, lisamisaeg
                FROM laulud WHERE id=?");
```

```
            $kask->bind_param("i", $_REQUEST["id"]);
            $kask->bind_result(
                $id, $pealkiri, $kommentaarid, $punktid, $lisamisaeg);
            $kask->execute();
            if($kask->fetch()){
               $pealkiri=htmlspecialchars($pealkiri);
                 $kommentaarid=nl2br(htmlspecialchars($kommentaarid));
               echo "
                   <h2>$pealkiri</h2>
                   <dl>
                     <dt>Punkte:</dt>
                       <dd>$punktid</dd>
                     <dt>Lisatud:</dt>
                       <dd>$lisamisaeg</dd>
                     <dt>Kommentaarid:</dt>
                       <dd>$kommentaarid</dd>
                   </dl>
                   <a href='?healaulu_id=$id'>Lisa punkt</a><br />
                   <form action='?'>
                          <input type='hidden'
                              name='uue_kommentaari_id' value='$id' />
                          <input type='text' name='uus_kommentaar' />
                          <input type='submit' value='Lisa kommentaar' />
                     </form>
                 ";
                 $kask->close();
            }
        }
      ?>
    <h1>Laulud</h1>
    <table>
        <?php
           $kask=$yhendus->prepare(
               "SELECT id, pealkiri, punktid FROM laulud WHERE avalik=1");
             $kask->bind_result($id, $pealkiri, $punktid);
             $kask->execute();
             while($kask->fetch()){
                $pealkiri=htmlspecialchars($pealkiri);
                echo "<tr>
                    <td><a href='?id=$id'>$pealkiri</a></td>
                    <td>$punktid</td>
                  </tr>";
             }
        ?>
      </table>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

Algul avaneb lehel laulude loetelu nagu tavaliselt, igaühele taha kirjutatud selle laulu punktide arv.

Laulud

Mutionu	9
Mutionu pidu	8
Muumioru lood	4
Rongisõit	1

Laulule vajutades saadetakse uuele lehepäringule kaasa selle laulu id (näha aadressiribal).

Saabunud id järgi küsitakse välja vastava laulu muud andmed ning näidatakse kasutajale. Samuti pannakse sinna siis viide punkti lisamiseks ning koht kommentaari sisestamiseks.

Mutionu pidu

Punkte:
8
Lisatud:
2012-10-10 13:30:30
Kommentaarid:

Lisa punkt

Lisa kommentaar

Laulud

Mutionu	9
Mutionu pidu	8
Muumioru lood	4
Rongisõit	1

Punktiviitele vajutamisel lisatakse punkt andmetabelisse ning seda näeb lehe järgmisel avamisel. Mutionu pidu on kaheksale punktile ühe juurde saanud.

Laulud

Mutionu	9
Mutionu pidu	9
Muumioru lood	4
Rongisõit	1

Ja kommentaarid jõuavad ka ilusti laulule külge.

Mutionu pidu

Punkte:
9
Lisatud:
2012-10-10 13:30:30
Kommentaarid:

Lisa punkt
Päris pikk laul | Lisa kommentaar

Laulud

Mutionu	9
Mutionu pidu	9
Muumioru lood	4
Rongisõit	1

Nõnda võib mõlema oskuse koos toimimist ühel lehel imetleda ning omale järgmiste rakenduste loomise juures alusnäidisena võtta.

Mutionu pidu

Punkte:
9
Lisatud:
2012-10-10 13:30:30
Kommentaarid:

Päris pikk laul

Lisa punkt
Lisa kommentaar

Laulud

Mutionu	9
Mutionu pidu	9
Muumioru lood	4
Rongisõit	1

Ülesandeid

- Tee näide läbi
- Pane ka ansamblite lehel kommenteerimine ja häälte andmine samale lehele
- Võimalda anda ka vastuhääli, st hääli vajutusega vähemaks võtta

Kohviautomaat

Andmetabeli kuju: (id, jooginimi, topsepakis, topsejuua)
Topside arv pakis näitab, mitu topsitäit saab juua ühe täitepakendi sisestamise peale.

- Loo tabel SQL-lausega. Lisa joogina kohv. Täitepaki suuruseks 50 topsi jagu pulbrit, algul masin tühi, juua pole midagi. Loo SQL-lause juua olevate topside arvu suurendamiseks täitepaki jagu. Käivita.
- Automaadi käivitatav leht vähendab juua olevate topside arvu ühe võrra. Vaataja leht näitab seda arvu.
- Automaat saab hakkama mitme joogiga (kohv, tee, kakao). Lehel näidatakse vaid neid jooke, millel on vähemasti üks tops juua. Joomise tulemusena vähendatakse vastava joogi olemasolevate topside loendurit. Halduslehel saab joodavate topside arvu kogust suurendada täitepaki jagu.

Jalgrattaeksami haldamise rakendus

Järgnevalt veidi pikem näide enamvähem tegeliku rakenduse veebiliidese kohta, kus sama sündmusega tegelevad mitu asjaosalist. Ehk siis tegemist abivahendina töö juures, kus muidu oleks päris palju sebimist, et vajalikud andmed õigel ajal õigesse kohta jõuaksid.

Üksinda päris lihtsalt rakendust tehes saab otsast vaikselt tegema hakata, veidi katsetada ning loodetavasti jõuabki mõne aja pärast kasutaja jaoks sobivale tulemusele. Kui aga tegijaid või kasutajaid mitu, või lihtsalt võtab rakenduse kokku panek rohkem aega kui paar päeva - sellisel juhul tuleb mõningane kavandamine ja plaanide ülesmärkimine kasuks. Siis rohkem lootust, et tulemus lõppkasutajale sobilik on ning ei pea nõnda palju tööd ringi tegema. Et pärast arendaja arvates rakenduse enam-vähem valmis saamist kulub vähemalt kolmandik tööd lõppviimistluse jaoks, see on tavapärane. Küllalt kergesti aga kipub juhtuma, et pärast esialgse lahenduse pealtnäha kõikide osade eraldi tööle hakkamist kulub veel kaks korda nõnda palju aega ja jõudu, et kuidagi töötavad lahendused võimalikult hästi töötavate mooduste vastu vahetada. Sest veebirakendusest on ju üldiselt kasu vaid siis, kui töö jõutakse kiiremini ja paremini teha võrrelduna pliiatsi ja paberi ning muude tavaliste vahendite abil tehtuna. Tavamooduseid on sageli aastakümneid kasutatud ja lihvitud. Veebilahenduse mugavaks saamiseks tuleb see kohandumisring ka ette võtta.

Üks levinud moodus rakenduste kavandamisel ja koostamisel on kirja panna või läbi käia järgmised osad:

- Rakenduse üldkirjelduse ülesmärkimine
- Kasutajate tegevuste kirjapanek üksikute kasutajalugude kaupa
- Veebilehtede struktuuri kirjapanek
- Lehtede ülesjoonistamine (paber)prototüüpidena, tegevuste läbimäng
- Lehtede kujundamine HTML-prototüüpidena
- Andmebaasiskeem
- Andmete ja kujunduse ühendamine (ehk esmapilgul põhiline töö)
- Lahenduse katsetamine tegijate hulgas
- Vajalike täienduste sisseviimine
- Lahenduse katsetamine sihtkohas, kohandamine, kuni võib tulemuse kõlblikuks lugeda.

Kavandamine

Jalgrattaeksami üldkirjeldus

Jalgrattaeksam koosneb kolmest etapist:

- Teooriaeksam
- Platsieksam
- Tänavasõidueksam

Teooriaeksami sooritamiseks on kümnest küsimusest vaja õigesti vastata vähemalt üheksa.

Platsieksamil tuleb reeglitepäraselt läbida slaalomirada ning näidata oma sõiduoskusi ringteel.

Tänavasõidueksamil tuleb eksamineeritavate grupil järjestikku sõita ees ja taga oleva eksamineerija vahel järgides liikluseeskirju.

Teooriaeksam peab olema sooritatud enne platsieksamit. Platsieksam peab olema sooritatud enne tänavasõidueksamit. Platsieksami osade läbimise järjekord pole tähtis.

Kasutajalood

Jalgrattalubade taotleja tuleb eksamile registreerimise laua juurde, esitab oma isikut tõendava dokumendi. Registreerija sisestab kasutaja andmed (lihtsamal juhul ees- ja perekonnanime) rakenduse kaudu infosüsteemi

Teoriaeksamiruumi sisenemisel kontrollitakse, et kohaletulnud on end eksamile registreerunud. Keda pole veel kirjas, suunatakse registreerimislaua juurde.

Teooriaeksami lahendanud taotlejatele sisestatakse nime juurde tema saadud punktide arv.

Platsieksamil on kaks kontrollpunkti, kummaski eraldi kontrollija koos sisestusseadmega. Kontrollijad näevad vaid neid nimesid, kes on registreeritud ning kel on kogutud teooriaeksamist vähemalt üheksa punkti. Kontrollija saab rakenduses määrata, kas eksamineeritav sai oma ülesandega hakkama (vastavalt siis slaalomisõiduga või ringteesõiduga vastavalt kontrollpunktile).

Tänavasõidueksami inspektorid näevad infosüsteemis vaid neid lubade taotlejaid, kel on läbitud mõlemad platsieksami kontrollpunktid. Vaid neid saavad nad tänavasõidueksamile lubada. Tänavasõidueksami lõpus märgivad inspektorid, et kellel tänavasõidueksam õnnestus, kellel mitte.

Lubade väljastamise laua töötajal on võimalik näha kõigi eksamineeritavate seisu. Kel pilt olemas, sellele väljastatakse luba ning määratakse sellega eksam lõpetatuks.

Rakenduse lehed

Jalgrattalubade taotleja registreerimine
  Sisestatakse taotleja ees- ja perekonnanimi. Andmed talletatakse kirjena tabelisse.

Teooriaeksam
  Nähakse loetelu registreeritud osalejatest, kes pole veel teooriaeksamil tulemust saanud.

  Eksamiülesande lahendanu nime juurde saab kirjutada tulemuse.

Slaalom
  Näha on teooriaeksami läbinud eksamineeritavad, kel veel pole kirjas tulemust slaalomi kontrollpunkti kohta. Saab soorituse määrata kas õnnestunuks või ebaõnnestunuks.

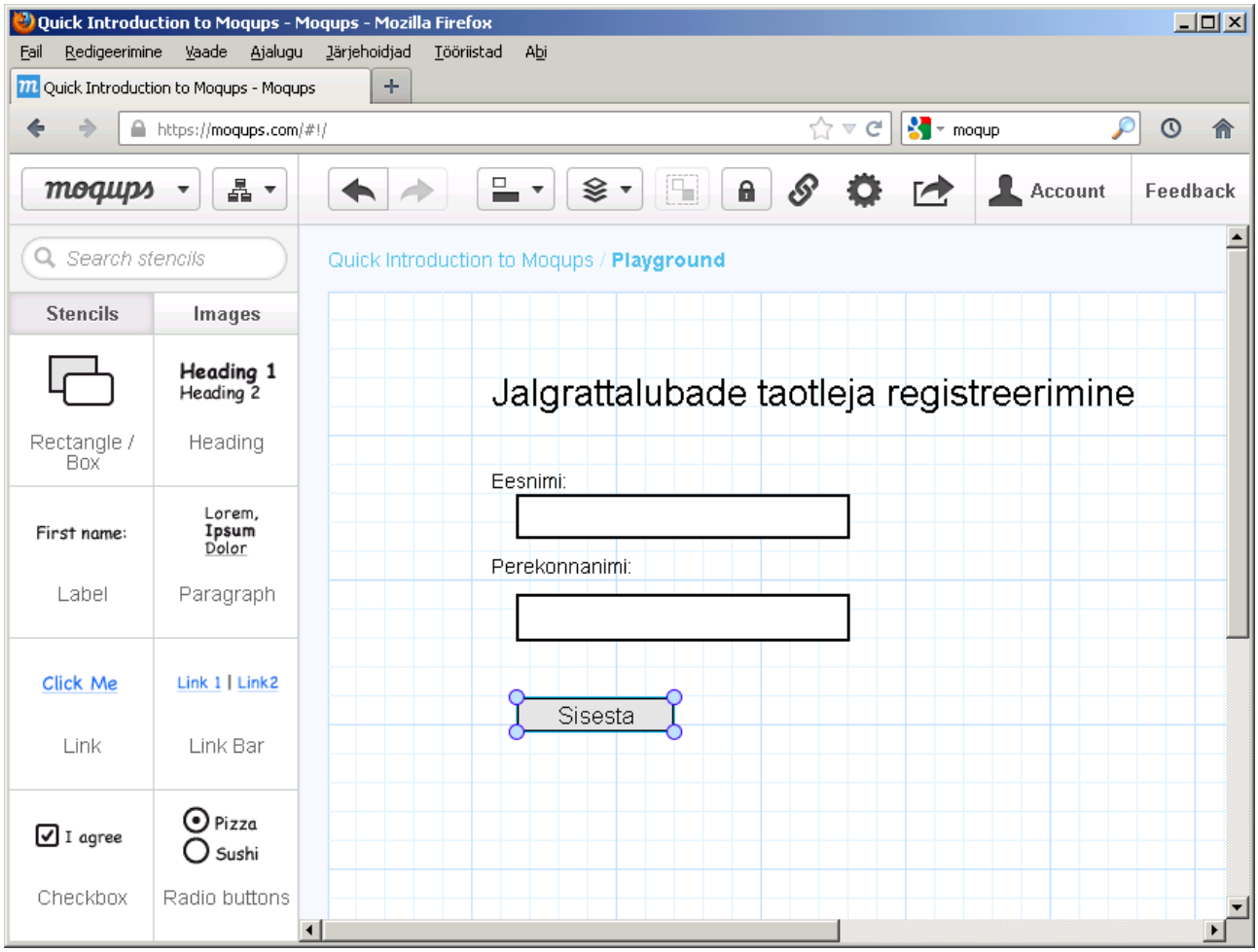
Ringtee
Näha on teooriaeksami läbinud eksamineeritavad, kel pole veel kirjas tulemust ringtee kontrollpunkti kohta. Saab soorituse määrata kas õnnestunuks või ebaõnnestunuks.

Tänavasõidueksam
Näha on nimekiri eksamineeritavatest, kes on läbinud platsieksami (ehk siis slaalomi ja ringtee kontrollpunkti) ning pole veel kirja saanud tulemust tänavasõidueksamil. Iga eksamineeritava kohta saab määrata tulemuse kas õnnestunuks või ebaõnnestunuks.

Vormistamise leht
Näha on kõikide osalejate tulemused. Kel kõik etapid korras, saab loa kätte ja see pannakse kirja.

Lehtede joonised

Järgmisena on viisakas valmis joonistada üksikud vaated. Olgu siis pastapliiatsi ja paberi abil, lihtsa joonistusprogrammiga või mõnd mockup-tööriista kasutades.



Jalgrattalubade teooriaeksam

Eesnimi	Perekonnanimi	Tulemus	
Juku	Tamm		Salvesta
Kati	Tamm		Salvesta

Slaalom

Juku	Tamm	Õnnestus	Ebaõnnestus
Kati	Tamm	Õnnestus	Ebaõnnestus
Mati	Tamm	Õnnestus	Ebaõnnestus

Ringtee

Juku	Tamm	Õnnestus	Ebaõnnestus
Kati	Tamm	Õnnestus	Ebaõnnestus
Mati	Tamm	Õnnestus	Ebaõnnestus

Tänavasõit

Juku	Tamm	Õnnestus	Ebaõnnestus
Kati	Tamm	Õnnestus	Ebaõnnestus
Mati	Tamm	Õnnestus	Ebaõnnestus

Vormistamine

Eesnimi	Perekonnanimi	Teooriaeksam	Slaalom	Ringtee	Tänavasõit	Lubade väljastus
Juku	Juurikas	9	korras	korras	.	.
Kati	Tamm	10	korras	korras	korras	Väljastatud
Mati	Kask	10	korras	korras	korras	Vormista

Edasi tasub läbi mängida kasutajalood jälgides, et kas ja kui mugavalt on neid joonistatud vaateid kasutades võimalik süsteem läbida. Piisavalt lihtsalt loodud kavanditele saab kergesti kommentaare juurde lisada. Ning kui katsetamise käigus selgub, et mõni muu lahendus oleks parem, siis on suhteliselt hõlbus ka vana skeem uuega asendada.

Andmebaasiskeem

Rakenduse loomise võimaluste juures on tähtis osa andmebaasiskeemil. Kasutada õnnestub enamasti vaid neid andmeid, mis baasis olemas. Siin näites piirdutakse andmete hoidmisel ühe tabeliga. Kuid erisuguste andmete lisandumisel võib tabelite arv kergesti kasvama hakata. Lühidalt kirja panduna on tabel järgnevate tulpadega.

```
jalgrattaeksam(id, eesnimi, perekonnanimi, teooriatulemus, slaalom, ringtee,
t2nav, luba)
```

Andmebaasiprogrammile tabeli loomisel arusaadavaks SQL-lauseks on

```
CREATE TABLE jalgrattaeksam(
  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  eesnimi VARCHAR(30),
  perekonnanimi VARCHAR(30),
  teooriatulemus INT DEFAULT -1,
  slaalom INT DEFAULT -1,
  ringtee INT DEFAULT -1,
  t2nav INT DEFAULT -1,
  luba  INT DEFAULT -1
);
```

-1 tähistab sisestamata tulemust
teooriatulemuse puhul 0-10 tähistab saadud punktide arvu
muude tulpade puhul
1 tähistab, et sooritus õnnestus
2 tähistab, et sooritus ei õnnestunud

Rakenduse käiguks tarvilikud SQL-laused.

Lehtede tööks vajalikud andmete küsimise, lisamise, muutmise ja kustutamise laused on hea enne eraldi välja kirjutada ning lehtede joonistega võrrelda. Siis paistab välja, et milliseid andmeid kust saadakse ning kas kõik vajalik on olemas.

Taotleja registreerimine

```
INSERT INTO jalgrattaeksam (eesnimi, perekonnanimi) VALUES ('Juku', 'Juurikas');
INSERT INTO jalgrattaeksam (eesnimi, perekonnanimi) VALUES ('Kati', 'Tamm');
INSERT INTO jalgrattaeksam (eesnimi, perekonnanimi) VALUES ('Mati', 'Kask');
```

Teooriaeksamil loetelu eksamineeritavatest, kes pole veel teooriaeksamil tulemust saanud.

```
SELECT id, eesnimi, perekonnanimi FROM jalgrattaeksam WHERE teooriatulemus=-1;
```

Teooriaeksami tulemuse sisestamine

```
UPDATE jalgrattaeksam SET teooriatulemus=9  WHERE id=1;
UPDATE jalgrattaeksam SET teooriatulemus=10 WHERE id=2;
UPDATE jalgrattaeksam SET teooriatulemus=10 WHERE id=3;
```

Loetelu eksamineeritavatest, kes saavad slaalomipunktis oma oskusi näidata

```
SELECT id, eesnimi, perekonnanimi FROM jalgrattaeksam
    WHERE teooriatulemus>=9 AND slaalom=-1;
```

Slaalomipunkti edukalt läbituks märkimine

```
UPDATE jalgrattaeksam SET slaalom=1 WHERE id=2;
```

Loetelu eksamineeritavatest, kes saavad ringteepunktis oma oskusi näidata

```
SELECT id, eesnimi, perekonnanimi FROM jalgrattaeksam
    WHERE teooriatulemus>=9 AND ringtee=-1;
```

Ringteepunkti edukalt läbituks märkimine

```
UPDATE jalgrattaeksam SET ringtee=1 WHERE id=2;
```

Loetelu eksamineeritavatest, kel õigus tänavasõidueksamile minna

```
SELECT id, eesnimi, perekonnanimi FROM jalgrattaeksam
  WHERE slaalom=1 AND ringtee=1 AND t2nav=-1;
```

Hetkeandmete väljund:

```
+----+---------+---------------+
| id | eesnimi | perekonnanimi |
+----+---------+---------------+
|  2 | Kati    | Tamm          |
+----+---------+---------------+
```

Tänavasõidueksami määramine sooritatuks:

```
UPDATE jalgrattaeksam SET t2nav=1 WHERE id=2;
```

Lubade laua juures kõigi tulemuste nägemine:

```
SELECT id, eesnimi, perekonnanimi, teooriatulemus, slaalom, ringtee, t2nav, luba
FROM jalgrattaeksam;
```

Lubade väljastamise märkimine:

```
UPDATE jalgrattaeksam SET luba=1 WHERE id=2;
```

Veebilehtede loomine

Pärast selliste eeltööde läbi viimist on valmivast rakendusest juba mõnevõrra lähem ettekujutus olemas ning võib loota, et kokkupandav rakendus ka kasutatav on. Ehkki juhtub küllalt sageli, et pärast esialgse versiooni tööle panekut tuleb ta veel mitme koha pealt ümber teha enne, kui kasutajad tulemusega rahul on. Kasutajalugude, skeemide ja esialgsete SQL-lausete kohendamine on aga algul tunduvalt lihtsam kui valmiskujundusega lahenduse pidev ümbermängimine. Samuti eriti suuremate lahenduste puhul ei pruugi kogu kavand sugugi kohe korraga pähe mahtuda. Üksikuid vaateid ja lõike aga julgeb ikka eraldi katsetada ning nende pealt jõuab vaikselt ka suurema lahenduse kokku panna.

konf.php

Suurema lahenduse puhul on seaded hea panna eraldi konfiguratsioonifaili. Praegusel juhul tulevad siia andmebaasiühenduse andmed. Kuid hea on olemasolu korral ka kõiksugu muud lisandused ühte koondada. Nagu ka lõpus kommentaar ütleb, siis juhul, kui PHP-fail midagi otse ekraanile väljastama ei pea, siis on lubatud ja soovitatav PHP lõpumärk ?> ära jätta.

```
<?php
 $baasiaadress="localhost";
 $baasikasutaja="juku";
 $baasiparool="kala";
 $baasinimi="jukubaas";
 $yhendus=new mysqli($baasiaadress, $baasikasutaja, $baasiparool, $baasinimi);

 //PHP lõpumärki pole vaja, et kogemata midagi välja ei trükitaks
```

registreerimine.php

Edasi võib vaikselt hakata vaadetele vastavaid lehti tegema. Mõnikord sobib ühte faili kokku mitu vaadet. Või mõnel korral on mugav üks vaade ehitada mitmest failist. Kuid alustuseks võib enamasti üks-ühele seotus sobida. Kuni lehed on suhteliselt iseseisvad ning neid seob andmete poolest andmebaas, siis saab samale andmestikule mugavasti lehti eraldi külge ehitada ning need ei hakka üksteist segama.

Eksami käigu poolest esimene tarvilik leht on registreerimisvorm. Andmed sisestatakse veebilehel, talletatakse baasi INSERT-lause abil. Äramärkimist väärib nime lisamise teate kuvamine veebilehel. Uuesti lehte avama kutsuva header-käskluse aadressirea parameetrina antakse vajalik teade, mis siis lehe avamisel välja kuvatakse.

```
header("Location: $_SERVER[PHP_SELF]?lisatudeesnimi=$_REQUEST[eesnimi]");
```

registreerimine.php

```
<?php
  require_once("konf.php");
```

```
  if(isSet($_REQUEST["sisestusnupp"])){
    $kask=$yhendus->prepare(
        "INSERT INTO jalgrattaeksam(eesnimi, perekonnanimi) VALUES (?, ?)");
      $kask->bind_param("ss", $_REQUEST["eesnimi"], $_REQUEST["perekonnanimi"]);
      $kask->execute();
      $yhendus->close();
      header("Location: $_SERVER[PHP_SELF]?lisatudeesnimi=$_REQUEST[eesnimi]");
      exit();
  }
?>
<!doctype html>
<html>
  <head>
     <title>Kasutaja registreerimine</title>
  </head>
  <body>
    <h1>Registreerimine</h1>
    <?php
        if(isSet($_REQUEST["lisatudeesnimi"])){
         echo "Lisati $_REQUEST[lisatudeesnimi]";
        }
    ?>
      <form action="?">
        <dl>
          <dt>Eesnimi:</dt>
            <dd><input type="text" name="eesnimi" /></dd>
          <dt>Perekonnanimi:</dt>
            <dd><input type="text" name="perekonnanimi" /></dd>
          <dt><input type="submit" name="sisestusnupp" value="sisesta" /></dt>
        </dl>
      </form>
  </body>
</html>
```

Tulemusena koht, kus nimi sisse kirjutada ning pärast sisestusnupule vajutamist jõuab see andmebaasitabelisse.

Pärast vajutust ilmuval uuel lehel näeb ametnik kinnitust oma saadetud andmete sisestamise kohta.

teooriaeksam.php

Järgmisena on vaja registreerunud eksamituppa kutsuda. Kõigepealt näha, et kes üldse tulemas on. Ning pärast ülesannete lahendamist ja kontrollimist tuleb märkida, millised tulemused saadi. Selleks siis kõigepealt SELECT-lause nende registreerunute leidmiseks, kel veel teooriaeksam tegemata (punktide arv -1). Ning pärast igaühe juures UPDATE-lause, mis osalise punktid paika määrab. Et igaühe andmed saaks mugavasti eraldi saata, selleks on iga nime taga olev sisestusväli eraldi vormis, kus pannakse varjatud väljana kaasa ka vastava registreerunu id-number.

```
<?php
  require_once("konf.php");
  if(!empty($_REQUEST["teooriatulemus"])){
    $kask=$yhendus->prepare(
        "UPDATE jalgrattaeksam SET teooriatulemus=? WHERE id=?");
      $kask->bind_param("ii", $_REQUEST["teooriatulemus"], $_REQUEST["id"]);
      $kask->execute();
  }
  $kask=$yhendus->prepare("SELECT id, eesnimi, perekonnanimi
     FROM jalgrattaeksam WHERE teooriatulemus=-1");
  $kask->bind_result($id, $eesnimi, $perekonnanimi);
  $kask->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Teooriaeksam</title>
  </head>
  <body>
    <table>
        <?php
          while($kask->fetch()){
              echo "
                <tr>
                    <td>$eesnimi</td>
                    <td>$perekonnanimi</td>
                    <td><form action=''>
                           <input type='hidden' name='id' value='$id' />
                               <input type='text' name='teooriatulemus' />
```

```
                                <input type='submit' value='Sisesta tulemus' />
                            </form>
                        </td>
                    </tr>
                ";
            }
        ?>
      </table>
  </body>
</html>
```

slaalom.php

Platsieksami juures üheks punktiks on slaalomisõit. Sinna pääsevad need registreerunud, kes kogusid teooriaeksamil vähemasti 9 punkti ning kes pole veel slaalomitulemust kirja saanud. Tulemus tähendaks seda, et slaalomisõit on kas korras või ebaõnnestunud. Kasutajaliides on inspektorile võimalikult lihtne, et seda suudaks kergesti ka platsil kaasas oleva miniseadme pealt vaadata.

```
<?php
  require_once("konf.php");
  if(!empty($_REQUEST["korras_id"])){
    $kask=$yhendus->prepare(
        "UPDATE jalgrattaeksam SET slaalom=1 WHERE id=?");
      $kask->bind_param("i", $_REQUEST["korras_id"]);
      $kask->execute();
  }
  if(!empty($_REQUEST["vigane_id"])){
    $kask=$yhendus->prepare(
        "UPDATE jalgrattaeksam SET slaalom=2 WHERE id=?");
      $kask->bind_param("i", $_REQUEST["vigane_id"]);
      $kask->execute();
  }
  $kask=$yhendus->prepare("SELECT id, eesnimi, perekonnanimi
     FROM jalgrattaeksam WHERE teooriatulemus>=9 AND slaalom=-1");
  $kask->bind_result($id, $eesnimi, $perekonnanimi);
  $kask->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Slaalom</title>
  </head>
  <body>
    <h1>Slaalom</h1>
    <table>
        <?php
          while($kask->fetch()){
              echo "
                <tr>
                    <td>$eesnimi</td>
                    <td>$perekonnanimi</td>
                    <td>
                      <a href='?korras_id=$id'>Korras</a>
```

```
                    <a href='?vigane_id=$id'>Ebaõnnestunud</a>
                  </td>
                </tr>
              ";
            }
        ?>
      </table>
  </body>
</html>
```

Slaalom

Siim Tamm Korras Ebaõnnestunud
Kati Kask Korras Ebaõnnestunud

ringtee.php

Ringteeharjutus on korraldusliku poole pealt slaalomiga sarnane. Kuna platsiharjutuste läbimise järjekord pole tähtis, siis ringteeharjutusele pääsemiseks on sama tingimus kui slaalomi puhul - ehk siis peab teooriaeksam tehtud olema.

```
<?php
  require_once("konf.php");
  if(!empty($_REQUEST["korras_id"])){
    $kask=$yhendus->prepare(
        "UPDATE jalgrattaeksam SET ringtee=1 WHERE id=?");
      $kask->bind_param("i", $_REQUEST["korras_id"]);
      $kask->execute();
  }
  if(!empty($_REQUEST["vigane_id"])){
    $kask=$yhendus->prepare(
        "UPDATE jalgrattaeksam SET ringtee=2 WHERE id=?");
      $kask->bind_param("i", $_REQUEST["vigane_id"]);
      $kask->execute();
  }
  $kask=$yhendus->prepare("SELECT id, eesnimi, perekonnanimi
     FROM jalgrattaeksam WHERE teooriatulemus>=9 AND ringtee=-1");
  $kask->bind_result($id, $eesnimi, $perekonnanimi);
  $kask->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Ringtee</title>
  </head>
  <body>
    <h1>Ringtee</h1>
    <table>
        <?php
          while($kask->fetch()){
              echo "
```

```
                <tr>
                    <td>$eesnimi</td>
                    <td>$perekonnanimi</td>
                    <td>
                      <a href='?korras_id=$id'>Korras</a>
                      <a href='?vigane_id=$id'>Ebaõnnestunud</a>
                    </td>
                  </tr>
              ";
            }
        ?>
      </table>
  </body>
</html>
```

t2nav.php

Tänavasõidule lastakse siis, kui mõlemad platsiharjutused edukalt läbitud. Seetõttu ka vastav pikem kontroll sealjuures.

```
  $kask=$yhendus->prepare("SELECT id, eesnimi, perekonnanimi
     FROM jalgrattaeksam WHERE slaalom=1 AND ringtee=1 AND t2nav=-1");
```

Muu osa aga eelmistele failidele suhteliselt sarnane.

```
<?php
  require_once("konf.php");
  if(!empty($_REQUEST["korras_id"])){
    $kask=$yhendus->prepare(
        "UPDATE jalgrattaeksam SET t2nav=1 WHERE id=?");
      $kask->bind_param("i", $_REQUEST["korras_id"]);
      $kask->execute();
  }
  if(!empty($_REQUEST["vigane_id"])){
    $kask=$yhendus->prepare(
        "UPDATE jalgrattaeksam SET t2nav=2 WHERE id=?");
      $kask->bind_param("i", $_REQUEST["vigane_id"]);
      $kask->execute();
  }
  $kask=$yhendus->prepare("SELECT id, eesnimi, perekonnanimi
     FROM jalgrattaeksam WHERE slaalom=1 AND ringtee=1 AND t2nav=-1");
  $kask->bind_result($id, $eesnimi, $perekonnanimi);
  $kask->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Tänavasõit</title>
  </head>
  <body>
    <h1>Tänavasõit</h1>
    <table>
        <?php
          while($kask->fetch()){
              echo "
                <tr>
                    <td>$eesnimi</td>
                    <td>$perekonnanimi</td>
                    <td>
                      <a href='?korras_id=$id'>Korras</a>
                      <a href='?vigane_id=$id'>Ebaõnnestunud</a>
                    </td>
                  </tr>
```

```
            ";
          }
        ?>
      </table>
  </body>
</html>
```

Tänavasõit

Juku Juurikas Korras Ebaõnnestunud
Mati Kask Korras Ebaõnnestunud

lubadeleht.php

Lõpetuslaua juures tasub kõiki eelnevaid andmeid näha. Et kui mõnel osalejal tekib küsimusi, et kuhu maani ta välja jõudis, siis seal on paras koht tulemusi vaadata. Lihtsamal juhul kuvatakse tabeli sisu veebilehele. Ning piisavalt teadlik asjaosaline juba teab sealt välja vaadata, et milline number mida tähendab.

```
<?php
  require_once("konf.php");
  $kask=$yhendus->prepare(
    "SELECT id, eesnimi, perekonnanimi, teooriatulemus,
            slaalom, ringtee, t2nav, luba FROM jalgrattaeksam;");
  $kask->bind_result($id, $eesnimi, $perekonnanimi, $teooriatulemus,
        $slaalom, $ringtee, $t2nav, $luba);
  $kask->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Lõpetamine</title>
  </head>
  <body>
    <h1>Lõpetamine</h1>
      <table>
        <?php
          while($kask->fetch()){
               echo "
                 <tr>
                     <td>$eesnimi</td>
                     <td>$perekonnanimi</td>
                     <td>$teooriatulemus</td>
                     <td>$slaalom</td>
                     <td>$ringtee</td>
                     <td>$t2nav</td>
                     <td>$luba</td>
                  </tr>
               ";
          }
```

```
        ?>
      </table>
  </body>
</html>
```

Lõpetamine

Juku	Juurikas	9	1	1	-1	-1
Kati	Tamm	10	1	1	1	1
Mati	Kask	10	1	1	-1	-1
Mari	Tamm	9	1	1	1	1
Toomas	Naaber	10	1	-1	-1	-1
toomas	naaber	9	2	-1	-1	-1
Jaan	Tamm	9	2	-1	-1	-1
Siim	Tamm	10	2	-1	-1	-1
Mati	Kask	9	1	-1	-1	-1
Siim	Tamm	9	1	-1	-1	-1
Kati	Kask	10	-1	-1	-1	-1

lubadeleht.php ilusamalt

Lõpetaja tööülesannete hulka kuulub ka lubade väljastamine - järelikult ka see toimetus sobib siinse lehe juurde. Ning esimesi päevi tööl oleval ametnikul on mugavam vaadata selgesõnalisi seletusi, kellel mis on korras ja mis mitte. Selle tarvis lisati väike alamprogramm asenduste tarbeks - numbrile vastavalt antakse välja sobiv tekst. Nõnda näeb allolev leht juba märgatavalt ametlikum välja.

```
<?php
  require_once("konf.php");
  if(!empty($_REQUEST["vormistamine_id"])){
    $kask=$yhendus->prepare(
        "UPDATE jalgrattaeksam SET luba=1 WHERE id=?");
      $kask->bind_param("i", $_REQUEST["vormistamine_id"]);
      $kask->execute();
  }
  $kask=$yhendus->prepare(
    "SELECT id, eesnimi, perekonnanimi, teooriatulemus,
           slaalom, ringtee, t2nav, luba FROM jalgrattaeksam;");
  $kask->bind_result($id, $eesnimi, $perekonnanimi, $teooriatulemus,
        $slaalom, $ringtee, $t2nav, $luba);
  $kask->execute();

  function asenda($nr){
    if($nr==-1){return ".";} //tegemata
    if($nr== 1){return "korras";}
    if($nr== 2){return "ebaõnnestunud";}
```

```
      return "Tundmatu number";
  }
?>
<!doctype html>
<html>
  <head>
    <title>Lõpetamine</title>
  </head>
  <body>
    <h1>Lõpetamine</h1>
      <table>
        <tr>
          <th>Eesnimi</th>
          <th>Perekonnanimi</th>
          <th>Teooriaeksam</th>
          <th>Slaalom</th>
          <th>Ringtee</th>
          <th>Tänavasõit</th>
          <th>Lubade väljastus</th>
        </tr>
        <?php
          while($kask->fetch()){
               $asendatud_slaalom=asenda($slaalom);
               $asendatud_ringtee=asenda($ringtee);
               $asendatud_t2nav=asenda($t2nav);
               $loalahter=".";
               if($luba==1){$loalahter="Väljastatud";}
               if($luba==-1 and $t2nav==1){
                 $loalahter="<a href='?vormistamine_id=$id'>Vormista load</a>";
               }
               echo "
                 <tr>
                     <td>$eesnimi</td>
                     <td>$perekonnanimi</td>
                     <td>$teooriatulemus</td>
                     <td>$asendatud_slaalom</td>
                     <td>$asendatud_ringtee</td>
                     <td>$asendatud_t2nav</td>
                     <td>$loalahter</td>
                   </tr>
               ";
             }
        ?>
      </table>
  </body>
</html>
```

Lõpetamine

Eesnimi	Perekonnanimi	Teooriaeksam	Slaalom	Ringtee	Tänavasõit	Lubade väljastus
Juku	Juurikas	9	korras	korras	.	.
Kati	Tamm	10	korras	korras	korras	Väljastatud
Mati	Kask	10	korras	korras	.	.
Mari	Tamm	9	korras	korras	korras	Väljastatud
Toomas	Naaber	10	korras	.	.	.
toomas	naaber	9	ebaõnnestunud	.	.	.
Jaan	Tamm	9	ebaõnnestunud	.	.	.
Siim	Tamm	10	ebaõnnestunud	.	.	.
Mati	Kask	9	korras	.	.	.
Siim	Tamm	9	korras	.	.	.
Kati	Kask	10	korras	korras	korras	Vormista load

Oma lahenduse loomise ülesanded

Eelneva jalgrattaeksami lahenduse saab näiteks kõrvale võtta ning nüüd on paras aeg oma valitud teemal sarnane arendusprotsess läbi käia. Kui tegijal parajasti paremat ideed ei tule, siis heaks harjutuseks on näiteks pitsapoe tellimuste haldus. Kõigepealt paika rakenduse lihtne üldkirjeldus, et kõrvaltvaatajalt veidi lähemalt mõista oleks, millega tegemist. Siis saab ükshaaval paika panna tegevused ja rollid - kel mida põhjust ja õigus teha. Edasi on viisakas järjest kirja panna kõik võimalikud toimingud, mis veebirakenduses ühe pitsatellimuse juures on mõistlik ette võtta. Ka tasub näiteks mõelda sellele, kas kuidagi peaks rakendus teadma sellest, et esimene pannile pandud pitsa läks kõrbema ning uue valmimine võtab lisaaega.

Kui toimetused kirjas, siis tasub asuda vaadete välja joonistamisele. Esialgu kõik lehed lihtsate skeemidena. Nende peal on hea võimalikud olukorrad läbi mängida. Kui aga põhilised toimetused tunduvad töötama, siis võib mõne kujunduse ka põhjalikumalt korda teha ning ka HTMLina valmis kujundada - enne kui päris andmetega majandama ja pead valutama hakata.

Omaette rida on andmebaas välja mõelda - lihtsamal juhul sobiv andmetabel kokku panna, kus kõik ühe pitsa tellimisega seotud andmed sees. Ning siis lehekülgede kaupa SQL-laused valmis kirjutada, et näha, millised andmed kust tulevad ning kuhu lähevad. Sageli selgub selle käigus ka, et mõistlik on andmetabelit või kujunduslehti veidi kohandada, et tulemus mugavamalt saavutatav oleks.

Ja kui eeltööd tehtud, võib usinasti rakenduse kokkupaneku kallale asuda. Olgu siis üksipäini või koos sõbraliku seltskonnaga. Kui kujunduspildid ning andmebaasipäringud ees, siis võib juba üsna julgesti iga lehte eraldi teha ning pärast koos töötavat tulemust imetleda.

Suurema toimetuse kõrvale või asemele enesekontrolliks või kontrolltööks mõned ülesanded, mille kallal võiks siinse õppematerjali läbimise järel olla samuti jaksu jõudu katsuda.

Kohviautomaat

Andmetabeli kuju: (id, jooginimi, topsepakis, topsejuua)
Topside arv pakis näitab, mitu topsitäit saab juua ühe täitepakendi sisestamise peale.
* Loo tabel SQL-lausega. Lisa joogina kohv. Täitepaki suuruseks 50 topsi jagu pulbrit, algul masin tühi, juua pole midagi. Loo SQL-lause juua olevate topside arvu suurendamiseks täitepaki jagu. Käivita.
* Automaadi käivitatav leht vähendab juua olevate topside arvu ühe võrra. Vaataja leht näitab seda arvu.
* Automaat saab hakkama mitme joogiga (kohv, tee, kakao). Lehel näidatakse vaid neid jooke, millel on vähemasti üks tops juua. Joomise tulemusena vähendatakse vastava joogi olemasolevate topside loendurit. Halduslehel saab joodavate topside arvu kogust suurendada täitepaki jagu.

Viljaladu

Andmetabel koormad: (id, autonr, sisenemismass, lahkumismass)
* Koosta tabel. Loo SQL lause auto lisamiseks koos autonumbri ja sisenemismassiga. Loo lause valitud id-ga auto lahkumismassi määramiseks.
* Koosta veebileht, kus saab sisestada autonumbri ja sisenemismassi. Koosta teine veebileht, kus valitud autole saab määrata väljumismassi.
* Koosta leht, kus näha tulnud autod koos andmetega ning maha laetud koormate suurused. Autonumbrile vajutades näidatakse kõik selle auto reisid ning arvutatakse kokku toodud vilja kogus.

Toolivahendus

Andmetabel toolid: (id, toon, tellimiskogus, valminudkogus)
* Koosta tabel. Loo SQL-lause tellimuse sisestamiseks (toon tekstina, ja tellimiskogus). Valminuid algul 0. Loo käsklus valminud koguse suurendamiseks ühe võrra.
* Koosta veebileht tellimuse sisestamiseks. Näita veebilehel toolide tabeli seisu. Loo leht tellimusele vastava valminud koguse suurendamiseks ühe võrra.
* Valminud kogust saab suurendada vaid tellimustel, kus kogus pole veel täis. Eraldi lehel näita valminud tellimusi. Arvuta kokku, mitu tooli on veel teha.

Autoveod

Andmetabel veod: (id, algus, ots, aeg, autonr, juht, valmis)
* Koosta tabel. Loo SQL-laused tellimuse sisestamiseks (algus, ots, aeg), tellimusele autonumbri määramiseks, tellimusele juhi nime määramiseks.
* Koosta leht veotellimuse sisestamiseks (algus- ja otspunkt, soovitav aeg). Koosta leht tellitud, kuid ilma juhita vedude nägemiseks. Võimalda veole määrata juht.
* Näita lehel vedusid, mille juht või autonumber määramata. Võimalda neid määrata. Juhi ja autonumbriga vedude puhul näita eraldi need veod, mis pole veel valmiks määratud, võimalda tehtuks määrata.

Arvutikomplektid

Andmetabel: arvutitellimused(id, kirjeldus, korpus, kuvar, pakitud). Viimased kolm neist arvud väärtusega 0 või 1.
* Koosta tabel. Loo SQL-lause kirjelduse sisestamiseks. Loo lause määramaks, et valitud tellimuse korpus on komplekteeritud.
* Koosta veebileht tellimuse sisestamiseks. Näita sisestatud tellimuse ja komplekteerimata korpusega tellimusi, võimalda määrata korpus komplekteerituks.
* Eraldi saab valmiks määrata korpuse ja kuvari. Pakkimislehel näeb vaid neid lehti, kus korpus ja kuvar olemas, aga komplekt veel pakkimata. Saab määrata pakituks. Luuakse statistikaleht, kus kirjas tellimuste arv ning mitu neist on lõpetatud.

Hirmude maja

Lõbustuspargis olevate õuduste maja külastajate haldamise rakendus. Majas olevate ohtlike atraktsioonide tõttu tuleb inimeste üle arvet pidada.

Andmetabel: hirmumaja:(id, eesnimi, sisenes, lahkus).
* Loo andmetabel. Loo SQL-lause kasutaja lisamiseks. Lause etteantud id-ga kasutaja sisenemiseks. Lause etteantud id-ga kasutaja lahkumise märkimiseks.
* Koosta veebileht pileti ostmiseks. Kasutaja kohta sisestatakse eesnimi. Võimalusel näidatakse lehele tekkival piletil kasutaja eesnime ja id-d ($yhendus->insert_id).
* Sisenemisukse juures oleval lehel märgitakse piletit näidanud id-ga inimene sisenenuks, väljumisukse juures oleval lehel lahkunuks. Loetelus saab näha hirmude maja sees olevate inimeste loetelu.

Aknaruloode tootmine

Andmetabel rulood (id, mustrinr, riievalmis, puuvalmis, pakitud)
Tellimisel sisestatakse mustri number. Vastavalt mustrinumbrile tuleb lõigata sobiv riie ning värvida sobiv puuosa. Kui mõlemad on olemas, saab nad ühendada ja teelesaatmiseks pakki panna.
* Loo SQL lause soovitud mustrinumbriga tellimuse lisamiseks. Lause määratud id-ga tellimuse riidelõikuse valmiks määramiseks. Lause valimis riide- ja puuosadega kuid veel pakkimata ruloode näitamiseks.
* Koosta veebileht tellimuse lisamiseks. Riideosakonna tööline näeb koos mustrinumbritega tellimusi, kus riideosa pole veel lõigatud. Saab töö tegemisel määrata need lõigatuks.
* Loo sarnane leht puuosa tegijate jaoks. Komplekteerijate lehel näha tellimused, mida saab pakkima hakata. Pakitud tellimused saab pakituks märkida. Klient saab oma tellimuse numbri sisestamisel näha oma tellimuse seisu.

Tantsuvõistlus

Hambotantsuvõistlusel tantsitakse piki tänavat. Sama paari hindab žürii iga liige oma lõigul. Andmetabel tantsupaarid (id, hinne1, hinne2, hinne3)
* Loo SQL lause tantsupaari sisestamiseks. Loo lause esimeses punktis hindamata paaride näitamiseks. Loo lause määratud id-ga paarile hinde määramiseks.
* Loo veebileht tantsupaari lisamiseks. Loo veebileht paaride näitamiseks, kes pole veel esimest hinnet saanud. Hindaja saab paarile hinde andmiseks vajutada sobivat tema taga olevat numbrit ühest viieni.
* Loo sarnased lehed teiste hindamislõikude tarbeks. Loo koondleht võistluse lõpetanud paaride tulemuste vaatamiseks, kus on näha ka iga paari keskmine hinne.

Suusahüppevõistlus

Andmetabel suusahyppajad (id, alustanud, kaugus, valmis).
* Loo SQL laused suusahüppaja lisamiseks, laskumise alustamise märkimiseks, alustanud ja veel lõpetamata hüppaja leidmiseks, talle kauguse määramiseks ning hüppealast väljumise määramiseks.
* Loo veebileht suusahüppajate lisamiseks. Ühtlasi on näha igaühe seis ja tulemus.
* Loo veebileht, millega määratakse laskumine alustatuks. Korraga tohib laskuma lubada vaid ühe hüppaja, muidu antakse veateade. Teised peavad olema selleks ajaks hüppealast lahkunud.
* Loo leht kauguse sisestamiseks. Kaugus kirjutatakse parasjagu hüppava võistleja juurde. Näita hüpanud võistlejate paremusjärjestust koos kohtadega kauguste järgi.