HTML és Rails
Gyakorlat

Kovács Gábor

2017. október 10.

A gyakorlat célja, hogy kialakítsuk a félév során megoldandó feladat képernyőit HTML-ben, ahol lehet Rails metódusok felhasználásával.

Az első lépés a webalkalmazásunk keretének kialakítása ezt a nézetek között az alkalmazásszintű nézetben, vagyis a `layouts/application.html.erb`-ben tehetjük meg. Rendezzük el úgy az oldalunkat, hogy legyen benne egy fejrész, egy központi rész, amely bal oldalon egy keskeny menüsávból áll, jobb oldalon a tartalomból, és egy lábrész. Az elrendezést `div`-ekkel valósítjuk meg, mindegyikhez egyedi `id`-t rendelve.

```
<div id="page">
  <div id="header"></div>
  <div id="main">
    <div id="menu"></div>
    <div id="body">
      <%= yield %>
    </div>
  </div>
  <div id="footer">
    Copyright, RoR 2017
  </div>
</div>
```

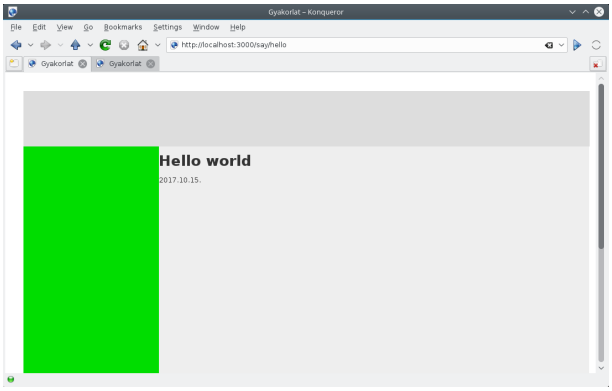
Második lépésként készítsük el az oldal stíluslapját, amivel ezek a helyükre kerülnek, és helyezzünk el benne minimális mennyiségű formázási információt. Az oldal legyen 1024 pixel széles. A fejrész legyen világosszürke és 100 képpont magas. A menüsávot a fejléc alatt helyezzük el, és az vízszintesen a szélesség 24%-át foglalja el, és legyen 600 pixel magas. Az oldal tartalmi része legyen 600 képpont magas, világosszürke háttérrel rendelkezzen, és a menütől jobbra helyezkedjen el a vízszintesen a szélesség 76%-át elfoglalva. A lábrészben a szöveget igazítsuk középre.

```
div#page {
    width: 1024px;
}

div#header {
    height: 100px;
    background-color: #ddd;
}

div#main {
    height: 600px;
}

div#menu {
    float: left;
    width: 24%;
    background: #0d0;
    height: 600px;
}

div#body {
    float: left;
    width: 76%;
    background-color: #eee;
    height: 600px;
}

div#footer {
    height: 100px;
    text-align: center;
    clear: both;
    background-color: #ddd;
}
```

Az így kialakított elrendezést például az 1. ábra mutatja.

Kétféle felhasználóra készülünk fel egyelőre, egy látogatóra és egy belépett felhasználóra, aki korábban keresztülment egy regisztrációs folyamaton. A látogató csak böngészhet, ugyanakkor regisztrálhat. A bejelentkezett felhasználó számára több funkciót is elérhetővé teszünk.

Kezdjük a látogató menüjével. Helyezzünk el a menüben egy a belépést lehetővé tevő formot! Ezt részleges rendereléssel tesszük meg. A formot tartalmazó fájl alkalmazás szinten kezeljük, ezért a `layouts` könyvtárban



1. ábra. Az oldal elrendezésének kialakítása

helyezzük el, így a be nem jelentkezett felhasználó bármelyik oldalon bejelentkezhet. A form tartalmazzon egy felhasználónévre utaló címkét és egy szövegbeviteli mezőt, valamint egy a jelszóra utaló címkét és jelszóbeviteli mezőt, továbbá egy Login feliratú nyomógombot. A formot a `form_tag` Rails helperrel valósítjuk meg, aminek első paramétere a formot kezelő URL, vagyis a form `action` attribútuma, illetve adjuk meg, hogy HTTP POST-tal kívánjuk elküldeni. A form mezőit rendre a `label_tag`, `text_field_tag`, `password_field_tag` és `submit_tag` helperekkel hozzuk létre, és a beviteli mezőket 16 karakter hosszúra korlátozzuk. A be nem jelentkezett felhasználónak tegyük lehetővé az elfelejtett jelszó visszaszerzését, ezt egy link hozzáadásával tesszük meg.

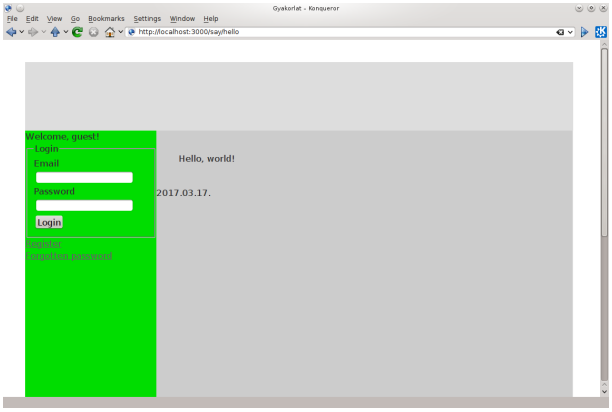
```
<h3>Hello guest!<h3>
<fieldset>
  <legend>Login</legend>
  <%= form_tag '/sessions/create', method: :post do %>
    <%= label_tag 'email', "Email" %><br/>
    <%= text_field_tag 'email', '', size: '20' %><br/>
    <%= label_tag 'password', "Password" %><br/>
    <%= password_field_tag 'password', '', size: '20' %><br
      />
    <%= submit_tag 'Login' %>
  <% end %>
</fieldset>

<%= link_to 'Register', '/users/new' %> <br/>
<%= link_to "Forgotten_password", '/users/forgotten' %>
```

Ezután a `menu` azonosítójú `div`-ben meghivatkozhatjuk ezt az oldalt. A

Rails konvenció szerint az aláhúzásjelet el kell hagynunk.

```
<div id="menu"><%= render 'layouts/menu' %></div>
```

A vendégfelhasználó menüjének megvalósítását a 2. ábra mutatja.



2. ábra. A vendégfelhasználó menüje

Modellezzük azt az esetet is, amikor egy felhasználó már bejelentkezett. Ezt egy, a `helpers/application_helper` állományban elhelyezendő saját helper metódussal tesszük meg. Itt egyelőre manuálisan állítjuk, hogy be van-e jelentkezve a felhasználó. A metódus értelemszerűen boolean visszatérési értékű.

```
module ApplicationHelper
  def logged_in?
    true
  end
end
```

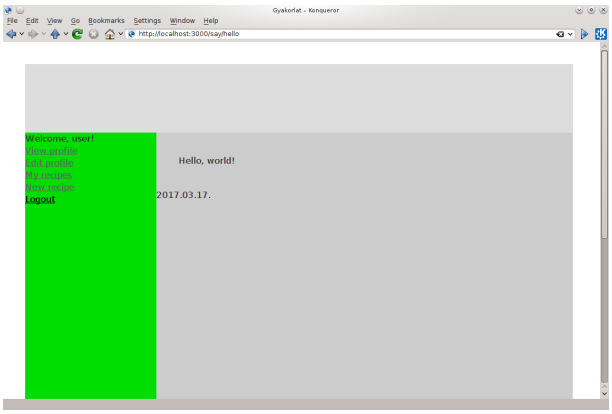
Ezt visszavezetve a keretbe a menüt megvalósító `menu`-ban a következő módosítást végezzük el. Így a helper módosításával be, illetve ki tudunk lépni az oldalról. „Jelentkeztessük” be és ki a felhasználót, hogy ellenőrizhessük, hogy a megfelelő menü jelenik-e meg a vendég és a felhasználó számára.

```
<% if logged_in? %>
  ...
<% else %>
  ...
<% end %>
```

A bejelentkezett felhasználó menüjét a vendéghez hasonlóan beágyazott nézettel hozzuk létre. Egyelőre hat akciót definiálunk: a sportszerek böngészését, a saját kölcsönzések böngészését, a saját kölcsönzéstörtének böngészését, profiloldal megtekintését, valamint szerkesztését és a kijelentkezést.

```
<h3>Hello valaki</h3>

<%= link_to "Browse items", "/items" %><br/>
<%= link_to "Borrowed items", "/items" %><br/>
<%= link_to "History", "/items" %><br/>
<%= link_to 'Edit profile', '/users/edit' %> <br/>
<%= link_to 'View profile', '/users/show' %> <br/>
<%= link_to "Logout", '/sessions/destroy' %>
```

A bejelentkezett felhasználó menüjének megvalósítását a 3. ábra mutatja.



3. ábra. A bejelentkezett felhasználó menüje

Nézzük meg a be nem lépett felhasználó regisztációs folyamatát! Az előző gyakorlat alkalmával már létrehoztuk a felhasználó modellünk kezdetleges változtatás, így arra már tudhatunk hivatkozni egy Rails formban, amely az MVC tervezési minta szerint szorosan kapcsolódik a nézethez. Hozzuk létre a felhasználó nézetét és fontosabb akcióit a következő paranccsal:

```
rails generate controller users new edit show forgotten
```

A parancs futtatásával létrejött az `users` kontroller és a hozzá kapcsolódó nézetek köztük az új felhasználó létrehozását lehetővé tevő `new`, a felhasználói profil szerkesztését megvalósító `edit`, a felhasználói adatait megjelenítő `show`, és az elfelejtett jelszó esetén az email címet elkérő `forgotten` nézet. Az tervezői kérdés, hogy az elfelejtett jelszó kezelését a felhasználók kontrollere

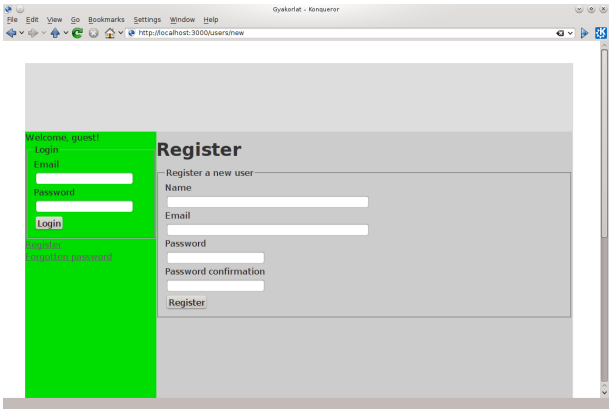
részének tekintjük, vagy önálló kontrollert hozunk létre számára. A gyakorlaton amellett döntöttünk, hogy az elfelejtett jelszó kerüljön a felhasználók kontrollerébe.

Hozzunk mindjárt létre a regisztrációs nézetet! Legyen egy címsorunk, amely elmondja a felhasználónak, hogy melyik oldalon van. Az esetleges hibaüzeneteknek tartsunk fenn helyet. Ezután egy `fieldset`-ben definiáljuk egy formot, amely ez esetben egy konkrét, létező modellhez van kötve. Ezt a `form_for` Rails helperrel tehetjük meg. Ennek első paramétere a modell neve szimbólum formájában, második paramétere a formhoz kötött akció, amely legyen a `users` kontroller `create` akciója, a harmadik paramétere a HTTP metódus, ami POST. A metódus blokkjának van egy paramétere a `form`, amin kereszül definiáljuk fogjuk az űrlap elemeit. Legyen az öt elem rendre a következő: egy 40 karakter széles, a felhasználó nevére vonatkozó szövegbeviteli mező a hozzá kapcsolódó címkével, egy 40 karakter széles, a felhasználó email címére vonatkozó szövegbeviteli mező a hozzá kapcsolódó címkével, egy 20 karakter széles, a felhasználó telefonszámára vonatkozó szövegbeviteli mező a hozzá kapcsolódó címkével, két darab 20 karakter széles jelszóbeviteli mező a hozzájuk kapcsolódó címkével, a két jelszómező eltérő azonosítóval rendelkezzék, az egyik prefixe `_confirmation`-re végződjék.

```
<h1>Registration</h1>
<%= flash[:notice] %>

<div>
  <fieldset>
    <legend>Register a new user</legend>
    <%= form_for @user, :url => { action: :create }, method
        : :post do |form| %>
      <div>
        <%= form.label :name %>
        <%= form.text_field :name, size: 40 %>
      </div>
      <div>
        <%= form.label :email %>
        <%= form.text_field :email, size: 40 %>
      </div>
      <div>
        <%= form.label :phone %>
        <%= form.text_field :phone, size: 20 %>
      </div>
      <div>
        <%= form.label :password %>
        <%= form.password_field :password, size: 20 %>
```

```
        </div>
        <div>
          <%= form.label :password_confirmation %>
          <%= form.password_field :password_confirmation,
              size: 20 %>
        </div>
        <div>
          <%= form.submit "Register" %>
        </div>
      <% end %>
    </fieldset>
</div>
```

A felhasználói regisztráció nézetét a 4. ábra mutatja.



4. ábra. A regisztráció nézete

A létrejött oldal HTML forrását tekintve a következőt látjuk. A formok mezőinek `name` és `id` attribútuma tartalmazza a modell nevét és a mező nevét. A név attribútum Ruby hash mintájára készült el, a modell nevének hash-ére hivatkozik a mező Rails forrásban megadott neve. Az általunk megadott mezőkön kívül létrejött két hidden mező is, amelyek a form használójának hitelesítését hivatottak ellenőrizni. A visszalépés itt JavaScripttel valósul meg. A forrást megtekintve láthatjuk, hogy a `:password_confirmation` szimbólumból a Rails automatikusan a *Password confirmation* szöveget állította elő. A stringek és a szimbólumok így ezen elv mentén felcserélhetők a form helperek argumentumlistájában.

A formok eseménykezelőihez a generate controller parancsunk nem generált útvonalakat, ezért azokat felvesszük a `config/routes.rb`-be. Jelenleg a nézetek kialakításához szingleton példányokkal dolgozunk, vagyis egyetlen

felhasználó adatait vagyunk képesek megjeleníteni. Később szükségünk lesz egy `:id` szimbólumra, ami képessé tesz minket a felhasználók megkülönböztetésére, és azonos nevű paraméterként jelentkezik majd a kontrollerben.

```
  get 'users/new'
  post 'users/create'

  get 'users/show'

  get 'users/edit'
  put 'users/update'

  get 'users/forgotten'
  post 'users/send_forgotten'
```

A form eseményét a Rails konvenció szerint a `create` kontroller metódus fogja kezelni. Ez még nem létezik, ezért definiáljuk azt egyelőre üres törzzsel.

A felhasználói profil szerkesztésének nézetében (`edit.html.erb`) található form szinte teljesen megegyezik az új felhasználót létrehozó formmal. A felhasználónév módosítását inaktívvá tehetjük, illetve az eseménykezelő kontroller akciót kell módosítanunk. A form eseményét a Rails konvenció szerint a `update` kontroller metódus fogja kezelni, erre létrehozzuk az útvonalat. Ez még nem létezik, ezért definiáljuk ezt is egyelőre szintén üres törzzsel. Ezen kívül a nézetben a feliratokat kell még átírnunk regisztrációról profil szerkesztésére.

```
<h1>Profile</h1>
<%= flash[:notice] %>

<div>
  <fieldset>
    <legend>Edit user profile</legend>
    <%= form_for @user, :url => { action: :update }, method
        : :put do |form| %>
      <div>
        <%= form.label :name %>
        <%= form.text_field :name, size: 40 %>
      </div>
      <div>
        <%= form.label :email %>
        <%= form.text_field :email, size: 40 %>
      </div>
      <div>
        <%= form.label :phone %>
```

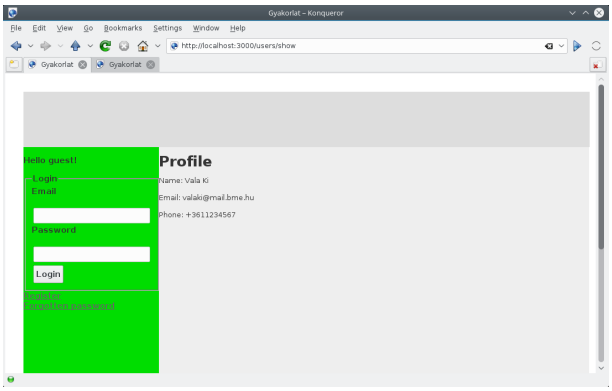
```
        <%= form.text_field :phone, size: 20 %>
      </div>
      <div>
        <%= form.label :password %>
        <%= form.password_field :password, size: 20 %>
      </div>
      <div>
        <%= form.label :password_confirmation %>
        <%= form.password_field :password_confirmation,
            size: 20 %>
      </div>
      <div>
        <%= form.submit "Modify" %>
      </div>
    <% end %>
  </fieldset>
</div>
```

Mivel a `form_for` Rails helper metódust használtuk a form létrehozására a `new` és az `edit` nézetekben, ezért szükséges a megfelelő kontroller akciókban a `@user` példányváltozó inicializálása. Ezeket egyelőre ne adatbázisból tegyük meg, hanem statikus tartalommal töltsük fel. Míg a `new` esetén a felhasználó még nem lézetik az adatbázisban, attribútumai inicializálatlanok, ezért elégséges egy frissen létrehozott példány használata, addig az `edit` esetén már ki kell töltenünk a struktúra mezőit beleértve az adatbázisbeli azonosító `id` attribútumot is.

```
class UsersController < ApplicationController
  def new
    @user = User.new
  end

  def show
    @user = User.new
    @user.name = "Vala Ki"
    @user.email = 'valaki@mail.bme.hu'
    @user.phone = '+3611234567'
  end

  def edit
    @user = User.new
    @user.name = "Vala Ki"
    @user.email = 'valaki@mail.bme.hu'
    @user.phone = '+3611234567'
```

```
  end

  def forgotten
    @user = User.new
  end
end
```

A felhasználói profiloldal szerkesztésének nézetét az 5. ábra mutatja. Láthatjuk, hogy a Rails automatikusan inicializálta a form mezőit, ahol a hozzájuk tartozó érték elérhető volt – a jelszó mezők kivételével.



5. ábra. A profiloldal nézete

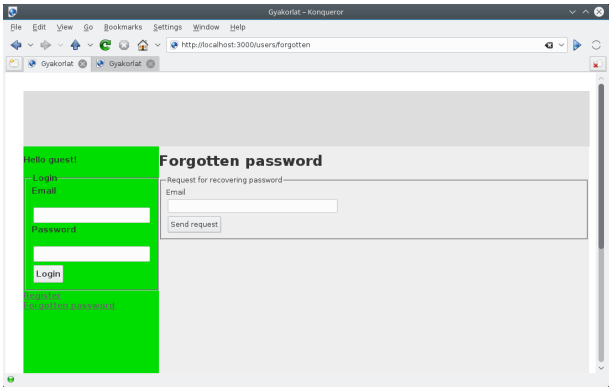
Ezután alakítsuk ki az elfejetett jelszó oldalt is. Itt egyszerűbb a formunk a beléptetésnél, csak az email címet tartalmazza.

```
<h1>Forgotten password</h1>
<%= flash[:notice] %>

<div>
  <fieldset>
    <legend>Request for recovering password</legend>
    <%= form_for @user, :url => { action: :send_forgotten
        }, method: :post do |form| %>
      <div>
        <%= form.label :email %>
        <%= form.text_field :email, size: 40 %>
      </div>
      <div>
        <%= form.submit "Send request" %>
      </div>
    <% end %>
```

```
  </fieldset>
</div>
```

Az elfelejtett jelszó kiküldését a form eseményét kezelő kontroller akció, a `send_forgotten` teszi majd meg, amit fel kell vennünk a kontroller osztályába egyelőre üres törzzsel.

Az elfelejtett jelszó nézetét a 6. ábra mutatja.



6. ábra. Az elfelejtett jelszó nézete

A sportszerkölcsönző portálunk központi elemei maguk a sportszerek, amelyeket az előző gyakorlaton `scaffold`-dal már létrehoztunk. A sportszerek kontrollerében négy nézetet létezik: `index`, `new`, `edit`, `show`. Az `index` nézetet átalakítjuk úgy, hogy az egyes sportszerek saját `show` nézetét beágyazzuk `<div>`-ek sorozatába. Két linket teszünk elérhetővé, az `index` nézetre vezető vissza linket, és az új eszköz létrehozása nézetre vezető linket.

```
<p id="notice"><%= notice %></p>

<h1>Items</h1>

<% @items.each do |item| %>
  <%= render 'show', locals: { item: item } %>
<% end %>

 <%= link_to 'Back', items_path %>|
 <%= link_to 'New Item', new_item_path %>
```

A beágyazott nézetet a `show` nézet klónozásával hozzuk létre, készítünk belőle egy `_show.html.erb` fájlnévvel egy másolatot, és abban a `@item` példányváltozókat lecseréljük a beágyazáskor átadott lokális változóra a `locals[:item]`-

re. Az `index` nézet eredeti táblázatából átmozgatjuk ide a megtekint, szerkeszt és töröl linkeket, valamint felveszünk két új linket is a kölcsönzésre és a visszaadásra. A nézet megjeleníti két, egymás melletti osztásban a sporteszközről készült képet, illetve a sporteszköz adatait: a nevét, a vonalkódját és a bérleti árát.

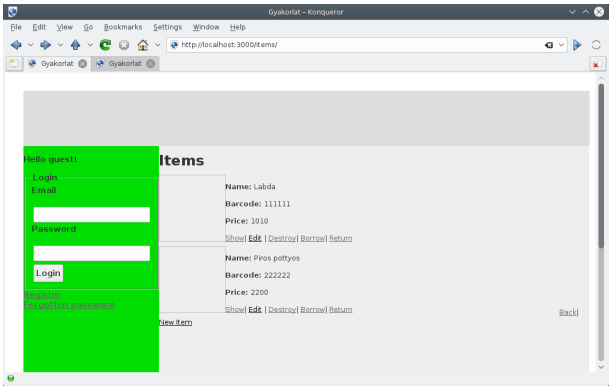
```
<div class="item">
  <div class="item-picture">
    <img src="" alt="Here come the image" width="120"
        height="120" />
  </div>
  <div class="item-data">
<p id="notice"><%= notice %></p>

<p>
  <strong>Name:</strong>
  <%= locals[:item].name %>
</p>

<p>
  <strong>Barcode:</strong>
  <%= locals[:item].barcode %>
</p>

<p>
  <strong>Price:</strong>
  <%= locals[:item].price %>
</p>

<%= link_to 'Show', locals[:item] %>|
<%= link_to 'Edit', edit_item_path(locals[:item]) %> |
<%= link_to 'Destroy', locals[:item], method: :delete, data
    : { confirm: 'Are you sure?' } %>|
<%= link_to 'Borrow', items_path %>|
<%= link_to 'Return', items_path %>
  </div>
</div>
```

Az elrendezés miatt módosítanunk kell a stíluslapot. A `show` nézet `div`-jeihez osztály attribútumot rendelünk. A kép lesz a bal oldalon, 120x120-as méretben, a szöveg szintén balra rendezve 600 pixel szélesen, és az egész egymás mellett helyezkedik el egyforma magassággal.

```
div.item-picture {
```

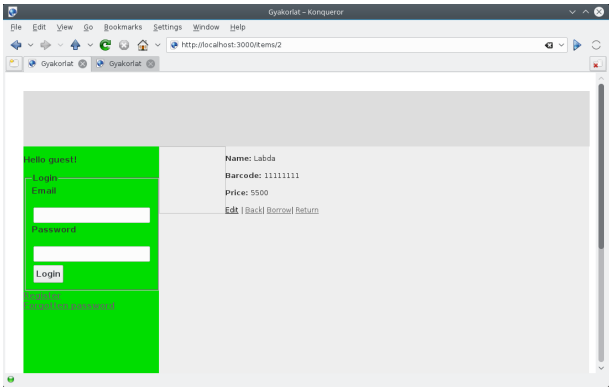
```
    float: left;
    width: 120px;
    height: 120px;
}

div.item-data {
    width: 600px;
    float: left;
}

div.item {
    display: table-row;
}
```

A sporteszközök index nézetét a 7, az egyes eszközök nézetét pedig a 8. ábra mutatja.

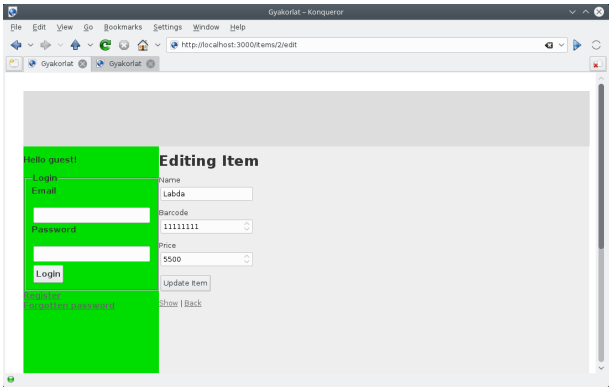


7. ábra. A sporteszközök index nézete

A sporteszközt létrehozó `new` nézet és a receptet szerkesztő `edit` nézet nagyon hasonló. Az oldal címsorában, az eseménykezelő akcióban és az eseményt kiváltó nyomógomb feliratában különböznek. A hozzájuk tartozó kontroller esetében a `new` metódusban egy üres objektumot kell inicializálnunk, az `edit` metódusban pedig az összes attribútumot, ez utóbbi tartalma egyezzen meg a `show` akciónál látottal. A szerkesztésének nézetét a 9. ábra mutatja.



8. ábra. Egyetlen sporteszközök részletező nézete



9. ábra. A sporteszköz szerkesztése