HTML és Rails
Gyakorlat

Kovács Gábor

2017. március 14.

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="body">
    <div id="menu"></div>
    <div id="main">
<%= 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 800 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.

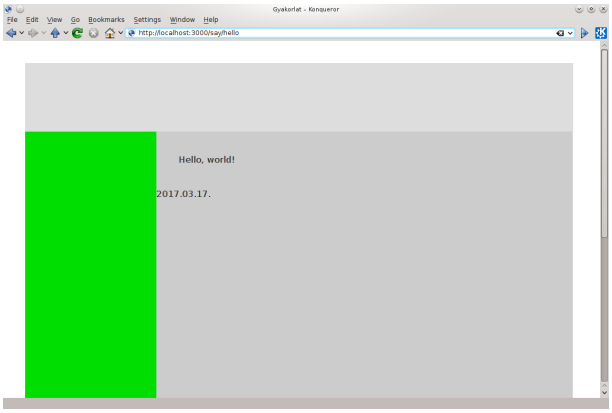
```
div#page {
    width: 800px;
```

```
}
div#header {
    height: 100px;
    background-color: #dddddd;
}
div#body {
    background-color: #eeeeee;
    height: 600px;
}
div#menu {
    background-color: #00dd00;
    width: 24%;
    float: left;
    height: 600px;
}
div#main {
    background-color: #cccccc;
    width: 76%;
    float: left;
    height: 600px;
}
div#footer {
    background-color: #dddddd;
    text-align: center;
    height: 100px;
    float: both;
}
```

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 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`,



1. ábra. Az oldal elrendezésének kialakítása

`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.

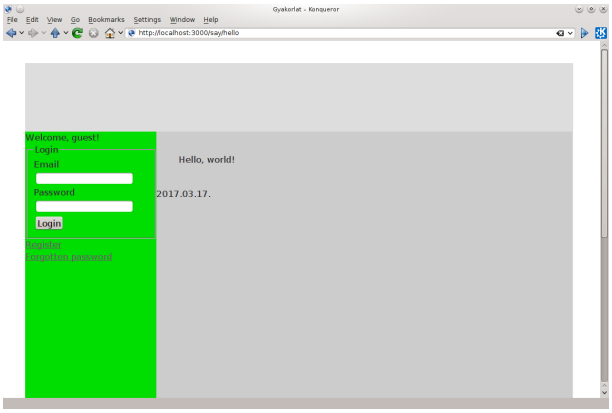
```
Welcome, guest!
<fieldset>
  <legend>Login</legend>
  <%= form_tag '/sessions/create', :method => :post do %>
    <%= label_tag :email, 'Email' %>
    <%= text_field_tag :email, '', :size => 18 %><br/>
    <%= label_tag :password, 'Password' %>
    <%= password_field_tag :password, '', :size => 18 %><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.

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-



2. ábra. A vendégfelhasználó menüje

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.

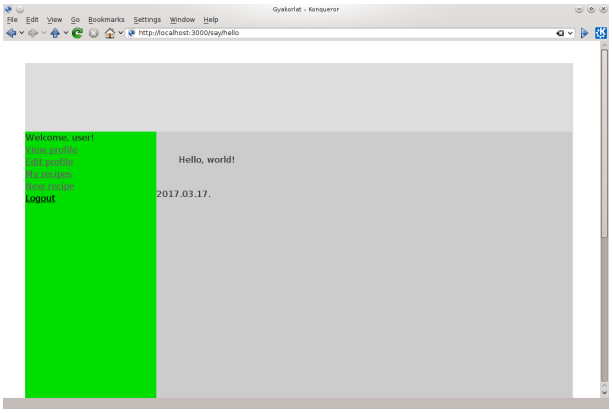
```
<% if logged_in? %>
  ...
<% else %>
  ...
<% end %>
```

A bejelentkezett felhasználó menüjét a vendéghez hasonlóan beágyazott nézettel hozzuk létre. Egyelőre öt akciót definiálunk: a profiloldal megtekintését, valamint szerkesztését, a saját receptek megtekintését, valamint új recept létrehozását és a kijelentkezést.

```
Welcome, user!<br/>
<%= link_to "View profile", user_path %><br/>
<%= link_to "Edit profile", profile_path %><br/>
<%= link_to "My recipes", recipes_path %><br/>
<%= link_to "New recipe", new_recipe_path %><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
```

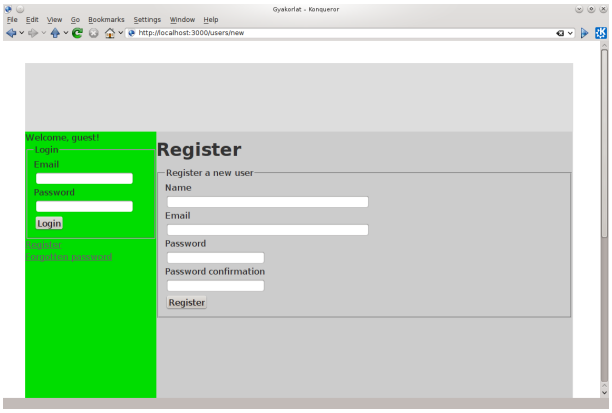
A parancs futtatásával létrejött az `users` kontroller és a hozzá kapcsolódó nézetek köztük az új felhasználó létrehozását lehetővé tevő `new`, a felhasználói profil szerkesztését megvalósító `edit`, a felhasználói adatait megjelenítő `show`, és az elfelejtett jelszó esetén az email címet elkérő `forgotten` 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 a négy 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, két darab 18 karakter széles jelszóbeviteli mező a hozzájuk kapcsolódó címkével, a két jelszómező eltérő azonosítóval rendelkezzék, az egyik prefixe `_confirmation`-re végződjék. Az oldal alján helyezzünk el egy visszalépés gombot azon felhasználóknak, akik véletletül tévedtek ide. A `:back` szimbólummal hivatkozott értékben a Rails a `HTTP_REFERER` HTTP fejléc elem vagy a Javascript `history` attribútuma alapján tárolja a megelőző HTTP kérésben használt URI-t.

```
<h1>Register</h1>
<div>
  <fieldset>
    <legend>Register a new user</legend>
    <%= form_for :user, url: '/users/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 :password %>
        <%= form.password_field :password, size: 18 %>
      </div>
      <div>
        <%= form.label :password_confirmation %>
        <%= form.password_field :password_confirmation,
            size: 18 %>
      </div>
      <%= form.submit 'Register' %>
    <% end %>
  </fieldset>
</div>

<%= link_to 'Back', :back %>
```

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 hashére 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', as: 'register', to: 'users#new'
post 'users/create', to: 'users#create'

get 'users/edit', as: 'profile', to: 'users#edit'
put 'users/update', to: 'users#update'

get 'users/show', as: 'user', to: 'users#show'

get 'users/forgotten', as: 'forgotten', to: 'users#
    forgotten'
```

```
post 'users/send_forgotten', to: 'users#send_forgotten'
```

Az `as` opció használatával az útvonalakat reprezentáló stringek karbantarthatóságán javíthatunk, így az csakis a `config/routes.rb` fájlban kell módosítanunk ezután, és nem az összes nézetben, ahol előfordul. A gyakorlatban minden útvonalra érdemes ilyen helper függvényt definiálnunk. A `'users/new'` stringet a `register_path` metódus állítja elő, ami az `as` opció és a `_path` vagy `_url` posztfix összefűzéséből adódik, és URL típusú függvényparaméterek helyett használható.

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>Edit user profile</h1>
<div>
  <fieldset>
    <legend>Edit user profile</legend>
    <%= form_for :user, url: '/users/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 :password %>
        <%= form.password_field :password, size: 18 %>
      </div>
      <div>
        <%= form.label :password_confirmation %>
        <%= form.password_field :password_confirmation,
            size: 18 %>
      </div>
```

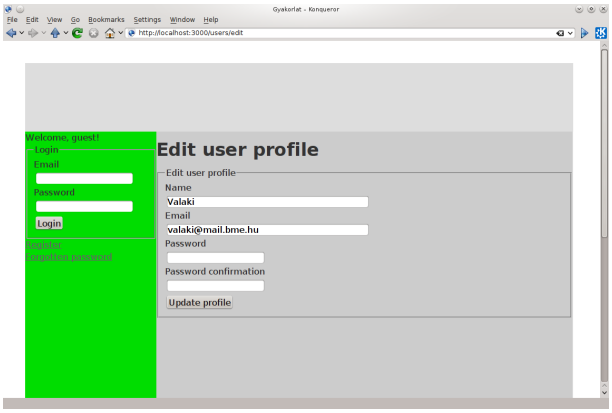
```
      <%= form.submit 'Update profile' %>
    <% end %>
  </fieldset>
</div>

<%= link_to 'Back', :back %>
```

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 show
    @user = User.new
    @user.name = 'Valaki'
    @user.email = 'valaki@mail.bme.hu'
  end

  def new
    @user = User.new
  end

  def create
  end

  def edit
    @user = User.new
    @user.name = 'Valaki'
    @user.email = 'valaki@mail.bme.hu'
  end

  def update
  end

  def forgotten
  end

  def send_forgotten
  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

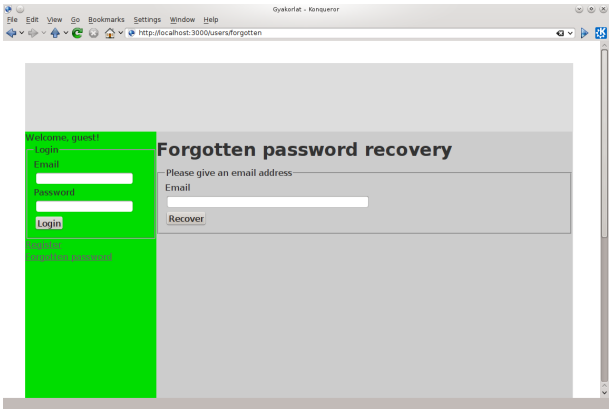
Ezután alakítsuk ki az elfejetett jelszó oldalt is. Itt egyszerűbb a formunk a beléptetésnél, csak az email címet tartalmazza.

```
<h1>Forgotten password recovery</h1>
<fieldset>
  <legend>Please give an email address</legend>
  <%= form_tag '/users/recover', method: :post do %>
    <%= label_tag :email %>
    <%= text_field_tag :email, '', size: 40 %><br/>
    <%= submit_tag 'Recover' %>
  <% end %>
</fieldset>

<%= link_to 'Back', :back %>
```

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. Ha a felhasználó meggondolná magát, és megsem kívánná elküldetni magának a jelszavát, egy `Back` feliratú linkkel biztosítjuk számára a lehetőséget az előző oldalra való visszatérésre.

Az elfelejtett jelszó nézetét a 6. ábra mutatja.

A receptmegosztó portálunk központi elemei maguk a receptek, amelyeket a létrehozó felhasználójuk szerkeszthet. Egy receptnek van egy string típusú



6. ábra. Az elfelejtett jelszó nézete

neve, egy szöveg típusú leírása, és a felhasználók táblára hivatkozó szerzője. Később tulajdonságként hozzáadjuk az elkészült ételről készült képet, a recept komplexitását és a recept elkészítési idejét. A receptek összetevőit kezelő nézeteket az előző gyakorlaton már létrehoztuk. Először definiáljuk a recept modelljét, és létrehozzuk hozzá a nézeteket.

```
rails g model Recipe name:string description:text user:
    references
rails g controller recipes new edit show index
rails db:migrate
```

A recepteket kezelő útvonalak módosítjuk a `router.rb`-ben:

```
  get 'recipes/new', as: 'new_recipe', to: 'recipes#new'
  post 'recipes/create', as: 'create_recipe', to: 'recipes#
     create'

  get 'recipes/edit', as: 'edit_recipe', to: 'recipes#edit'
  put 'recipes/update', as: 'update_recipe', to: 'recipes#
     update'

  get 'recipes/show', as: 'recipe', to: 'recipes#show'

  get 'recipes/index', as: 'recipes', to: 'recipes#index'
```

A receptek kontrollerében négy nézetet definiálunk: `index`, `new`, `edit`, `show`. Az `index` nézetben a recepteket egy sorszámozatlan listában jelenítjük meg úgy, hogy a lista elemei klikkelhetők, és átvisznek a recept részletező oldalára, melyet a `show` akció reprezentál.

```
<h1>Recipes</h1>
<ul>
  <% for recipe in @recipes do %>
  <li><%= link_to recipe.name, recipe_path %></li>
  <% end %>
</ul>
```

Az adokat a `@recipes` példányváltozóból vesszük, melyet a kontroller `index` akciójában inicializálunk.

```
  def index
    recipe1 = Recipe.new
    recipe1.name = "Kakaoscsiga"
    recipe2 = Recipe.new
    recipe2.name = "Nem tudom mi"
    @recipes = [ recipe1, recipe2 ]
  end
```

A recept részletező oldalának `show` nézetében megjelenítjük a recept nevét, alatta az összetevőinek listáját, azalatt pedig az elkészítésének leírását.

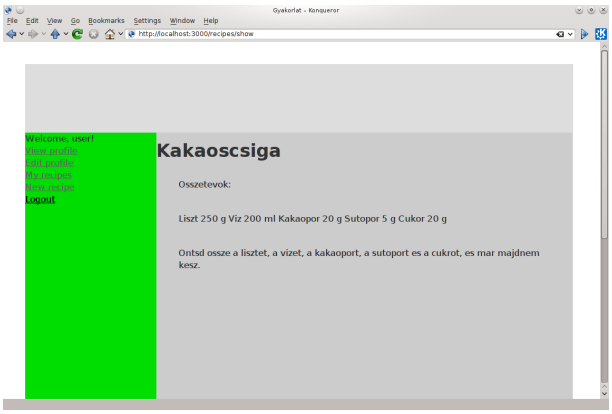
```
<h1><%= @recipe.name %></h1>
<p>Osszetevok:</p>
<p>
<% for i in @ingredients do %>
  <%= i.element %> <%= i.amount %> <%= i.unit %> <% print '
      \t' %>
<% end %>
</p>
<p><%= @recipe.description %></p>
```

A nézetben két példányváltozót használunk fel, a `@recipe`-t, amely a receptet reprezentálja, és a `@ingredients`-et, amely az összetevőket tartalmazza. Ezeket inicializálnunk kell a receptek kontroller `show` metódusában:

```
  def show
    @recipe = Recipe.new
    @recipe.name = 'Kakaoscsiga'
    @recipe.description = 'Ontsd ossze a lisztet, a vizet, a kakaoport, a sutop
ort es a cukrot, es mar majdnem kesz.'
    i1 = Ingredient.new
    i1.element = 'Liszt'
    i1.amount = 250
    i1.unit = 'g'
```

```
    i2= Ingredient.new
    i2.element = 'Viz'
    i2.amount = 200
    i2.unit = 'ml'
    i3 = Ingredient.new
    i3.element = 'Kakaopor'
    i3.amount = 20
    i3.unit = 'g'
    i4 = Ingredient.new
    i4.element = 'Sutopor'
    i4.amount = 5
    i4.unit = 'g'
    i5 = Ingredient.new
    i5.element = 'Cukor'
    i5.amount = 20
    i5.unit = 'g'
    @ingredients = [i1, i2, i3, i4, i5]
  end
```

A recept nézetét a 7. ábra mutatja.



7. ábra. A recept

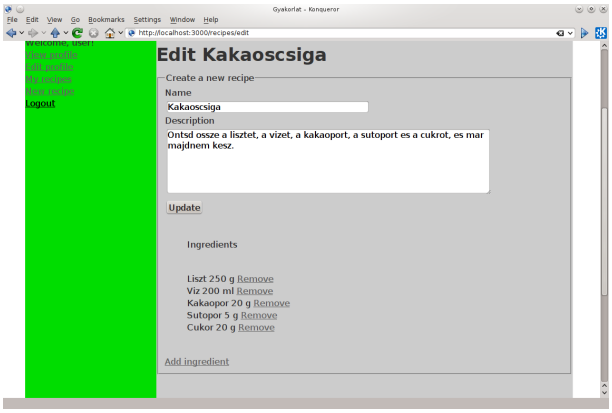
A receptet létrehozó `new` nézet és a receptet szerkesztő `edit` nézet nagyon hasonló. Az oldal címsorában, az eseménykezelő akcióban és az eseményt kiváltó nyomógomb feliratában különböznek. A hozzájuk tartozó kontroller esetében a `new` metódusban egy üres objektumot kell inicializálnunk, az `edit`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

Az `edit` nézeten tegyük szerkeszthetővé a recept attribútumait egy `form_for`

helperrel, alatta pedig listázzuk ki az összetevőket, és tegyünk az egyes összetevők mellé egy az összetevő eltávolítását lehetővé tevő linket. Az oldal alján pedig helyezzünk el egy linket, mellyel egy új összetevő hozzáadható ehhez a listához.

```
<h1>Edit <%= @recipe.name %></h1>
<fieldset>
  <legend>Create a new recipe</legend>
  <%= form_for :recipe, url: update_recipe_path, method: :
      put do |form| %>
    <div>
      <%= form.label :name %>
      <%= form.text_field :name, size: 40 %>
    </div>
    <div>
      <%= form.label :description %>
      <%= form.text_area :description, rows: 6, cols:66 %>
    </div>
    <%= form.submit "Update" %>
  <% end %>
  <p>Ingredients</p>
  <p>
  <% for i in @ingredients do %>
    <%= i.element %> <%= i.amount %> <%= i.unit %> <%=
        link_to "Remove", ingred
ient_path(2), method: :delete %> <br/>
  <% end %>
  </p>
  <%= link_to "Add ingredient", new_ingredient_path %>
</fieldset>
```

A recept szerkesztésének nézetét a 8. ábra mutatja.



8. ábra. A recept szerkesztése