HTML és Rails
Gyakorlat

Kovács Gábor

2020. október 13.

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="content">
      <%= yield %>
    </div>
  </div>
  <div id="footer">
    RoR, 2020
  </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. Az oldal központi része legyen 600 pixel magas. A menüsávot a központi részben, a fejléc alatt helyezzük el, és az vízszintesen a szélesség 24%-át foglalja el, magasságát a központi rész magassága definiál-

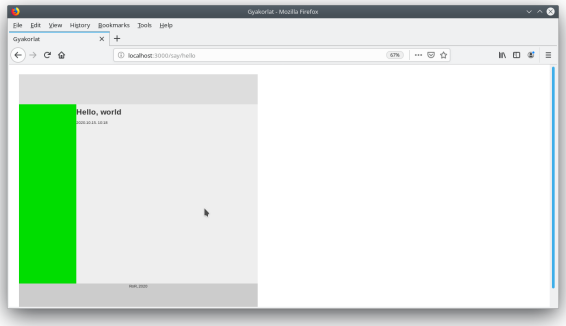
ja. Az oldal tartalmi része világosszürke háttérrel rendelkezzen, és a menütől jobbra helyezkedjen el a vízszintesen a szélesség 76%-át elfoglalva. A lábrészben a szöveget igazítsuk középre, és legyen az is 100 pixel magas, valamint a fejlécnél világosabb szürke színű.

```
div#page {
        width: 800px;
}
div#header {
        height: 100px;
        background-color: #ddd;
}
div#footer {
        height: 100px;
        background-color: #ccc;
        text-align: center;
        clear: both;
}
div#main {
        height: 600px;
}
div#menu {
        float: left;
        width: 24%;
        height: 100%;
        background-color: #0d0;
}
div#content {
        float: left;
        width: 76%;
        height: 100%;
        background-color: #eee;
}
```

Az így kialakított elrendezést például az 1. ábra mutatja.

Két felhasználótípusra készülünk fel egyelőre, egy látogatóra, egyfelhasználóra, az utóbbi korábban keresztülment egy regisztrációs folyamaton. A látogató csak böngészhet, bejelentkezhet és jelszóemlékeztetőt kérhet. A bejelentkezett felhasználók 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 bejelent-



1. ábra. Az oldal elrendezésének kialakítása

kezhet. A Rails konvenció szerint a részlegesen renderelt állományok neve aláhúzásjellel kezdődik. Legyen a fájlunk neve ezért `_guest.html.erb`! 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 18 karakter hosszúra korlátozzuk. A be nem jelentkezett felhasználónak tegyük lehetővé az elfelejtett jelszó visszaszerzését, ezt egy link hozzáadásával tesszük meg.

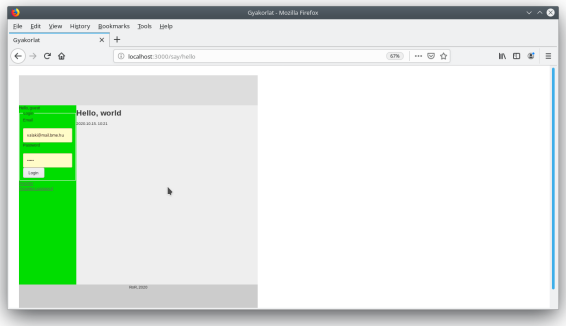
```
Hello, guest!

<fieldset>
  <legend>Login</legend>
  <%= form_tag '/sessions/create', method: :post do %>
    <%= label_tag 'email', 'Email' %><br/>
    <%= text_field_tag 'email', '', size: 14 %><br/>
    <%= label_tag 'password', 'Password' %><br/>
    <%= password_field_tag 'password', '', size: 14 %><br/>
    <%= submit_tag "Login" %>
  <% end %>
</fieldset>
<%= link_to "Register", '/users/new' %><br/>
<%= link_to "Forgotten password", '/users/forgotten' %><br/
  >
```

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

Rails konvenció szerint az aláhúzásjelet el kell hagynunk.

```
<div id="menu">
    <%= render '/layouts/user' %>
</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 `logged_in?`. 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.

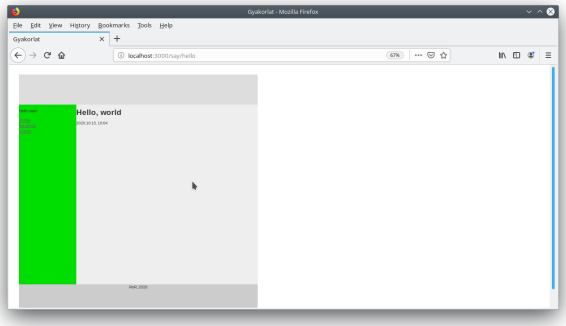
```
<div id="menu">
  <% if logged_in? %>
    <%#= render '/layouts/user' %>
  <% else %>
    <%#= render '/layouts/guest' %>
  <% end %>
```

```
</div>
```

A bejelentkezett felhasználó menüjét a vendéghez hasonlóan beágyazott nézettel hozzuk létre (`_user.html.erb`). Egyelőre három akciót definiálunk: a profiloldal megtekintését, az összes videó megtkintését, valamint a kijelentkezést.

```
Hello, student!<br>
<p>Hello user!</p>
<%= link_to "Profile", '/users/edit' %><br/>
<%= link_to "My videos", '/videos' %><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 létrehozásának 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:

```
kovacsg@debian:~/gyakorlat/app# rails g controller users
    new edit forgotten show
      create  app/controllers/users_controller.rb
       route  get 'users/new'
get 'users/edit'
get 'users/forgotten'
get 'users/show'
      invoke  erb
      create    app/views/users
      create    app/views/users/new.html.erb
```

```
      create    app/views/users/edit.html.erb
      create    app/views/users/forgotten.html.erb
      create    app/views/users/show.html.erb
      invoke  test_unit
      create    test/controllers/users_controller_test.rb
      invoke  helper
      create    app/helpers/users_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    scss
      create      app/assets/stylesheets/users.scss
```

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 felhasználó adatlapját megmutató `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 egy modell objektum vagy annak neve szimbólum formájában, második paramétere a formhoz kötött akció, amely legyen a `users` kontroller (ezt nem kell leírnunk, mert az új felhasználó létrehozása akció kontrollere ugyanaz) `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 20 karakter széles, a felhasználó nevére vonatkozó szövegbeviteli mező a hozzá kapcsolódó címkével, egy 20 karakter széles, a felhasználó email címére 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.

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

```
<h1>Registration</h1>

```

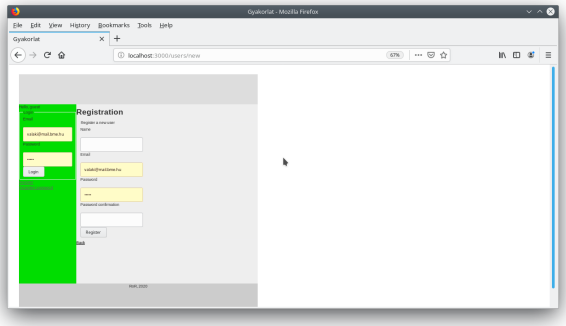
```
<fieldset>
  <legend>Register a new user</legend>
  <%= form_for @user, url: '/users/create', method: :post
      do |form| %>
    <div>
      <%= form.label :name %><br/>
      <%= form.text_field :name, size: 20 %>
    </div>
    <div>
      <%= form.label :email %><br/>
      <%= form.text_field :email, size: 20 %>
    </div>
    <div>
      <%= form.label :password %><br/>
      <%= form.password_field :name, size: 20 %>
    </div>
    <div>
      <%= form.label :password_confirmation %><br/>
      <%= form.password_field :name, size: 20 %>
    </div>
    <%= form.submit "Register" %>
  <% end %>
</fieldset>

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

Ahhoz, hogy az űrlap megjelenjen, a kontrollerben inicializánunk kell a `@user` példányváltozót.

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

A felhasználói regisztráció nézetét a 4. ábra mutatja.

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



4. ábra. A regisztráció nézete

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'
  get 'users/edit'
  get 'users/forgotten'
  get 'users/show'
  get 'say/hello'
```

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. Az email cím 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>

<fieldset>
```

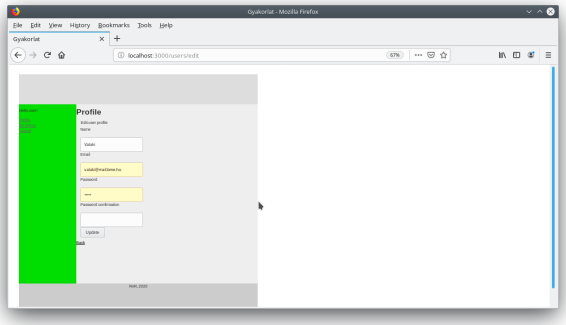
```
  <legend>Edit user profile</legend>
  <%= form_for @user, url: '/users/update', method: :put do
      |form| %>
    <div>
      <%= form.label :name %><br/>
      <%= form.text_field :name, size: 20 %>
    </div>
    <div>
      <%= form.label :email %><br/>
      <%= form.text_field :email, size: 20 %>
    </div>
    <div>
      <%= form.label :password %><br/>
      <%= form.password_field :password, size: 20 %>
    </div>
    <div>
      <%= form.label :password_confirmation %><br/>
      <%= form.password_field :password_confirmation, size:
          20 %>
    </div>
    <%= form.submit "Update" %>
  <% end %>
</fieldset>

<%= 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. A `show` akcióban is inicializáljuk a `@user` példányváltozót, a `index` akcióban pedig a `@users` példányváltozót, amely egy User típusú objektumokat tartalmazó tömb.

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

  def edit
    @user = User.new name: "Valaki", email: "valaki@mail.
```

```
        bme.hu"
  end

  def forgotten
  end

  def show
  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

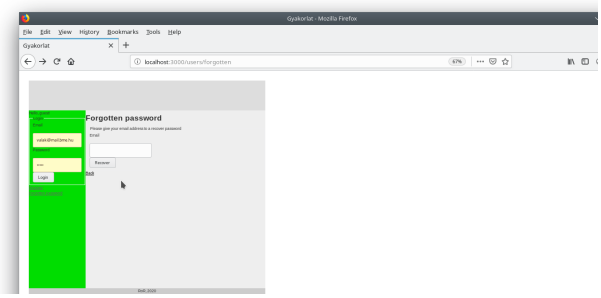
Ezután alakítsuk ki az elfejetett jelszó oldalt is. Itt egyszerűbb a formunk a beléptetésnél, csak az email címet tartalmazza.

```
<h1>Forgotten password</h1>

<fieldset>
  <legend>Please give your email address to a recover
      password</legend>
  <%= form_tag '/users/recover', method: :put do %>
    <div>
      <%= label_tag :email, "Email" %><br/>
      <%= text_field_tag :email, '', size: 20 %>
    </div>
    <%= 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.

Az elfelejtett jelszó nézetét a 6. ábra mutatja.



6. ábra. Az elfelejtett jelszó nézete

Hozzuk létre a videómegosztó portálunk videóira vonatkozó modellünket és a hozzá tartozó kontrollert egy paranccsal. Az `Video` modellünkben legyen egy string típusú, `title` nevű, és egy a tulajdonos felhasználóra vonatkozó referencia. A videók rendelkezni fognak tagekkel, kommentekkel és magát a videót tartalmazó állománnyal, ezeket a következő gyakorlatokon rendeljük majd hozzá.

```
kovacsg@debian:~/gyakorlat/> rails g scaffold video title:
    string user:references
       invoke  active_record
       create    db/migrate/20201013110647_create_videos.rb
       create    app/models/video.rb
       invoke    test_unit
       create      test/models/video_test.rb
       create      test/fixtures/videos.yml
       invoke  resource_route
        route    resources :videos
       invoke  scaffold_controller
       create    app/controllers/videos_controller.rb
       invoke    erb
       create      app/views/videos
       create      app/views/videos/index.html.erb
       create      app/views/videos/edit.html.erb
```

```
      create    app/views/videos/show.html.erb
      create    app/views/videos/new.html.erb
      create    app/views/videos/_form.html.erb
      invoke  test_unit
      create    test/controllers/videos_controller_test.
        rb
      create    test/system/videos_test.rb
      invoke  helper
      create    app/helpers/videos_helper.rb
      invoke    test_unit
      invoke  jbuilder
      create    app/views/videos/index.json.jbuilder
      create    app/views/videos/show.json.jbuilder
      create    app/views/videos/_video.json.jbuilder
      invoke  assets
      invoke  scss
      create    app/assets/stylesheets/videos.scss
      invoke  scss
      create  app/assets/stylesheets/scaffolds.scss
```

Létrejött egy `Video` modell, egy `VideosController` kontroller és a kapcsolódó nézetek: `new` és `edit` egy-egy nézet, amelyek a közös `_form` töredékben lévő formot használják a videó adatainak létrehozására, illetve módosítására. A `show` nézet a videó adatlapját mutatja, a `index` nézet pedig az elérhető videók mutatja egy táblázatban.

Hajtsuk végre a scaffold létrehozása során keletkezett migrációt.

```
kovacsg@debian:~/gyakorlat/app/views/users> rails db:
    migrate
== 20201013110647 CreateVideos: migrating
   ===========================================
-- create_table(:videos)
   -> 0.0399s
== 20201013110647 CreateVideos: migrated (0.0402s)
   ==================================
```

Ezután a böngészőben nyissuk meg a videók nézetet (`http://localhost:3000/videos`), próbáljuk ki a videó létrehozását, törlését, listázását. Nézzük meg, hogy létrejött-e a rekord az adatbázisban, nyissuk meg az adatbázis konzolt:

```
kovacsg@debian:~/gyakorlat/db/migrate# rails db
MariaDB [gyakorlat_development]> desc videos;
+-------------+--------------+------+-----+---------+----------------+
```

```
| Field      | Type            | Null | Key | Default | Extra
          |
+------------+-----------------+------+-----+---------+-------------------------------+

| id         | bigint(20)      | NO   | PRI | NULL    |
   auto_increment |
| title      | varchar(255) | YES  |     | NULL    |
                 |
| user_id    | bigint(20)      | NO   | MUL | NULL    |
                 |
| created_at | datetime(6)     | NO   |     | NULL    |
                 |
| updated_at | datetime(6)     | NO   |     | NULL    |
                 |
+------------+-----------------+------+-----+---------+-------------------------------+

5 rows in set (0.001 sec)

MariaDB [gyakorlat_development]> select * from videos;
+----+-------+---------+----------------------------+----------------------------+

| id | title | user_id | created_at                 |
   updated_at                 |
+----+-------+---------+----------------------------+----------------------------+

|  1 | Foci  |       2 | 2020-10-13 11:09:30.046025 |
   2020-10-13 11:09:30.046025 |
+----+-------+---------+----------------------------+----------------------------+

1 row in set (0.001 sec)
```

Mivel a Rails adatbáziskezelését még nem ismerjük, cseréljük le a `VideosController`-ben az automatikusan generált adatbázisműveleteket tartalmazó kódrészleteket statikus adatokra. Ezt két helyen kell megtennünk, az `index` akcióban, ahol a `@videos` példányváltozó videó példányokat tartalmazó tömbjét hozzuk létre, illetve a `before_action` helper függvény által kijelölt `set_video` azonosítójú metódusban, amely lefut mind a `show`, mind az `edit` nézet betöltésekor, itt csak a `@video` példányváltozóhoz kell egy videó példányt rendelnünk. A videókhoz rendelt felhasználó legyen az adatbázisban lévő egyetlen felhasználó.

```
class VideosController < ApplicationController
  before_action :set_video, only: [:show, :edit, :update, :
     destroy]
```

```
  def index
    @videos = []
    @videos << Video.new(id:1, title: 'Foci', user: User.
        find(2))
    @videos << Video.new(id:1, title: 'Foci2', user: User.
        find(2))
    @videos << Video.new(id:1, title: 'Foci3', user: User.
        find(2))

#     @videos = Video.all
  end

  def show
    @tags = ['SWE', "ENG"]
  end

  private
    def set_video
#       @video = Video.find(params[:id])
        @video = Video.new
        @video.id = 1
        @video.title = 'Foci'
        @video.user = User.find(2)
    end
end
```

A videóhoz rendelt tagek megjeleníthetősége végett a `show` akció számára inicializáljuk a tagek példányváltozóját. Hogy nézetekben felhasznált attribútumok hozzáférhezőek legyenek, a modell osztályban elérhetővé kell tennünk a hozzájuk tartozó settereket, gettereket mégha azokkal nem is végzünk adatbázis-műveleteket. Ilyenek a tagek és a forrás fájlja.

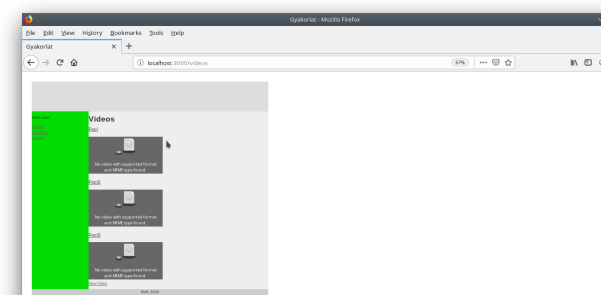
```
class Video < ApplicationRecord
  belongs_to :user
  attr_accessor :file

  def tags
    ['ENG', 'SWE']
  end
end
```

A videólista nézeten (`index`) előbb megsemmisítjük az automatikusan generált kódrészleteket, majd a videókat egymás alatt helyezzük el a `@videos`

példányváltozón iterálva. A későbbiek folyamán tördelni fogjuk őket, hogy három elférjen egymás alatt a kijelölt területen. A videókról megjelenítjük a címét, és a videót. A videó forrását a webszerver gyökérkönyvtárának számító `public` könyvtárban helyezzük el valahol.

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

<h1>Videos</h1>


<div class='video-list'>
    <% @videos.each do |video| %>
      <div class='video-box'>
          <h3><%= link_to video.title, '/videos/1' %></h3>
          <video width="250" controls class="video" id="
              video_<%= video.id %>">
            <source src="/videos/swe-eng.mp4" type="video/
                mp4" />
          </video>
      </div>
    <% end %>
</div>

<%= link_to 'New Video', new_video_path %>
```

Az videók listájának nézetét a 7. ábra mutatja.



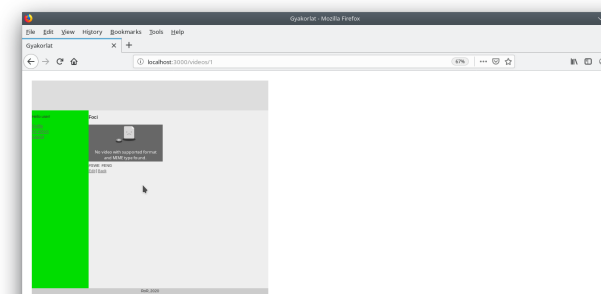
7. ábra. A videók listájának nézete

A videó címére kattintva átmehetünk a videó adatlapjára, ahol láthatjuk a videóhoz rendelt tageket is.

```
<H3><%= @video.title %></H3>
```

```
<div class="video-box">
  <video width="250" controls class="video">
    <source src="/videos/swe-eng.mp4" type="video/mp4" />
  </video>
</div>

<div class="tags">
  <% for tag in @tags do %>
    #<%=tag%> 
  <% end %>
</div>

<%= link_to 'Edit', edit_video_path(@video) %> |
<%= link_to 'Back', videos_path %>
```

Egy videó részletező nézetét a 8. ábra mutatja.



8. ábra. Egy videó részletező nézete

A videó létrehozása és szerkesztése oldalon beállítjuk a videó tagjeit egy legördülő menüből kiválasztva kijelölhetők a videóhoz rendelt tagek, valamint a videó forrásfájlja.

```
<%= form_with(model: video, local: true) do |form| %>
  <% if video.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(video.errors.count, "error") %>
          prohibited this video from being saved:</h2>

      <ul>
        <% video.errors.full_messages.each do |message| %>
          <li><%= message %></li>
        <% end %>
      </ul>
```

```
    </div>
  <% end %>

  <div class="field">
    <%= form.label :title %>
    <%= form.text_field :title %>
  </div>

  <div class="field">
    <%= form.label :user_id %>
    <%= form.text_field :user_id %>
  </div>

  <div class="field">
    <%= form.label :file %>
    <%= form.file_field :file %>
  </div>

  <div class="field">
    <%= form.label :tags %>
    <%= form.select :tags, [ "SWE", "ENG" ] %>
  </div>

  <div class="actions">
    <%= form.submit %>
  </div>
<% end %>
```

A videókat és a tageket összerendeljük, viszont előbb létrehozzuk a tagek modelljét és karbantartó nézeteit. A tagekről egyelőre csak magát a címkét jegyezzük meg.

```
rails g scaffold tags label:string
```

A migráció végrehajtása után a karbantartást az adminisztrátorra bízzuk úgy, hogy az admisztrátor képernyőkhöz saját layoutot rendelünk. A layout legyen a `layouts/application.html.erb` eredeti állapota. A `TagsController`-ben kell ezt a hozzárendelést megvalósítanunk, amennyiben az összes nézetre érvényesíteni akarjuk, egyébként, ha csak egy-egy nézetre akarjuk érvényesíteni, a kontroller megfelelő függvényében kell ezt megtennünk.

```
class TagsController < ApplicationController
  layout 'admin'
end
```