Exercises: Single Page Apps with AJAX, REST and Kinvey

Problems for exercise at the "JavaScript Applications" course @ SoftUni. Submit your solution in the SoftUni instance as homework. You can use this presentation to help you throughout this exercise.

1. Prodavachnik

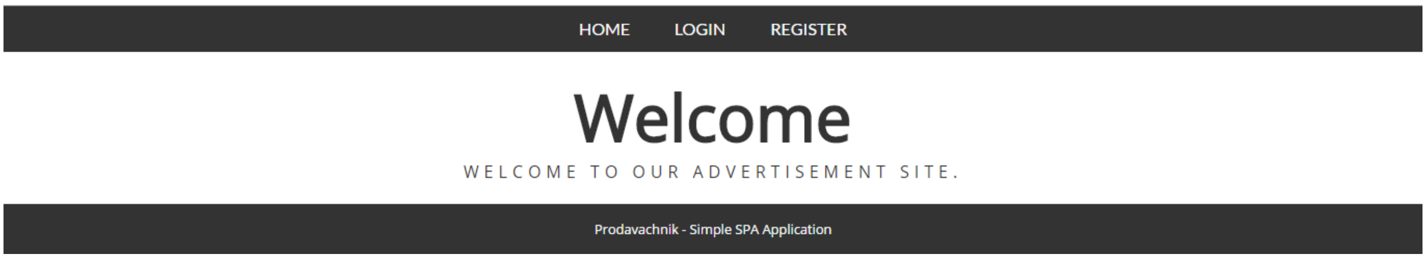
Your task is to implement a Web page that serves the functionally of an **advertisement site**, which specializes in **selling products/services**. The API will consist of creating, listing, editing and deleting of **advertisements**(**CRUD**). Every **advert** should contain the following properties:

- Title - String
- Description - String
- Publisher - String
- Date of Publishing – Date (format "**mm/dd/yyyy**")
- Price – **floating** point number, to the **second digit** after decimal separator.

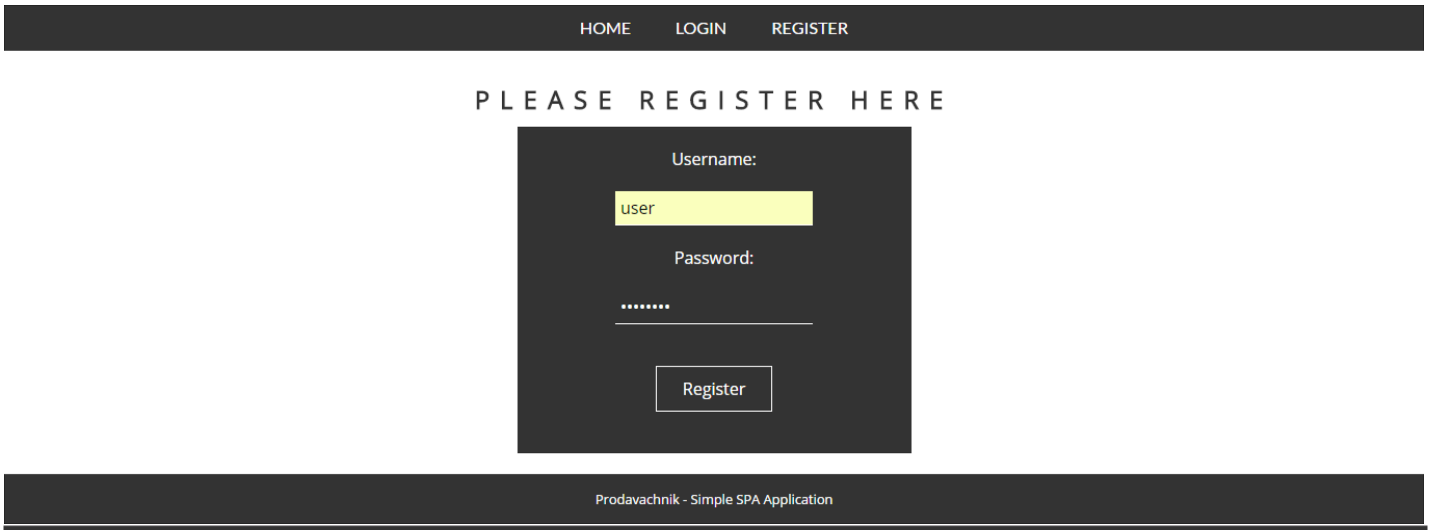
You can download the skeleton (HTML and CSS) here.

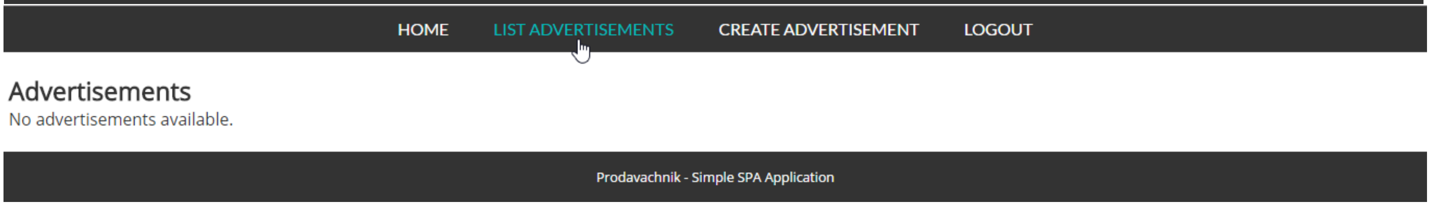
Screenshots

Home page:


HOME
LOGIN
REGISTER
Welcome
WELCOME TO OUR ADVERTISEMENT SITE.
Prodavachnik - Simple SPA Application

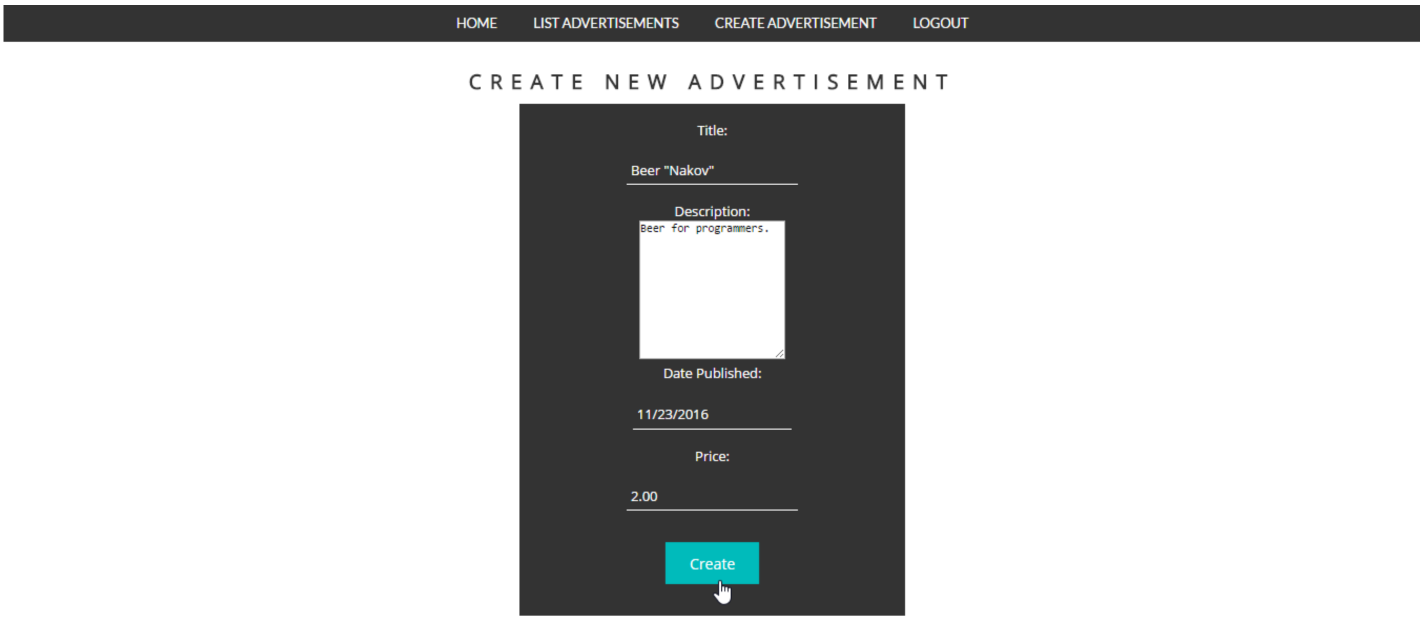
Registration page (Login page is identical):




HOME
LIST ADVERTISEMENTS
CREATE ADVERTISEMENT
LOGOUT
Advertisements
No advertisements available.
Prodavachnik - Simple SPA Application

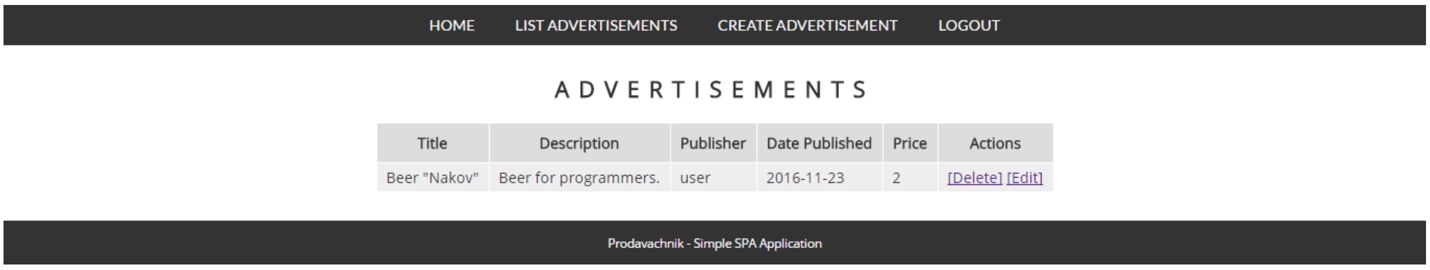
This is how the website should look like when there are no adverts:

Let’s create one (the view for editing an advert is identical):



Title:	Beer "Nakov"
Description:	Beer for programmers.
Date Published:	11/23/2016
Price:	2.00

And this is how it should be listed:



Title	Description	Publisher	Date Published	Price	Actions
Beer "Nakov"	Beer for programmers.	user	2016-11-23	2	[Delete] [Edit]

Prodavachnik - Simple SPA Application

The advertisement site should have the following functionality:

- Perform any **CRUD** operations over the advertisement entity.
  - Note that the **publisher** of the advertisement is the **currently** logged **user**.
  - Publisher can edit/delete his own adverts only.
- Have **Register/Login/Logout**.

Hints

- When **create** an **advertisement** the current logged user should be it's **publisher**. We can do it by making one request to the database (by user's **"_id"** property) and retrieve the whole **user** information. It may look something like that:

```
function createAdvert() {
    const kinveyAuthHeaders = {
        'Authorization': "Kinvey " + sessionStorage.getItem('authToken'),
    };

    const kinveyUserUrl =
        `${kinveyBaseUrl}user/${kinveyAppKey}/${sessionStorage.getItem('userId')}`;

    $.ajax({
        method: "GET",
        url: kinveyUserUrl,
        headers: kinveyAuthHeaders,
        success: afterPublisherRequest,
        error: showError
    });
```

  Once we got the user information we can use it in our "afterPublisherRequest" method:

```
function afterPublisherRequest(publisher) {
    let advertData = {

      publisher: publisher.username,
```

  And the rest is **parsing** the **information** from the form and sending **another request** – this time to **save** our **advert** in database.

  Note that the above method should be called when the form for **advert creation** is **submitted**.

- When **editing** an advert in which we don't want to change the **publisher** you can include it in the html, but mark it as "hidden", here is how the html might look:

```
<section id="viewEditAd" class="viewEditAd">
    <h1 class="titleForm">Edit existing advertisement</h1>
    <form id="formEditAd" class="form">
        <div><input type="hidden" name="id" required /></div>
        <div><input type="hidden" name="publisher" required /></div>
```

  What is left is to **initially set** the **value** when printing the edit form **and** then **parsing** the whole **data** when the form is **submitted**.

2. Extend "Prodavachnik" (Optional)

Your task is to extend the functionality of the above project (when completed).

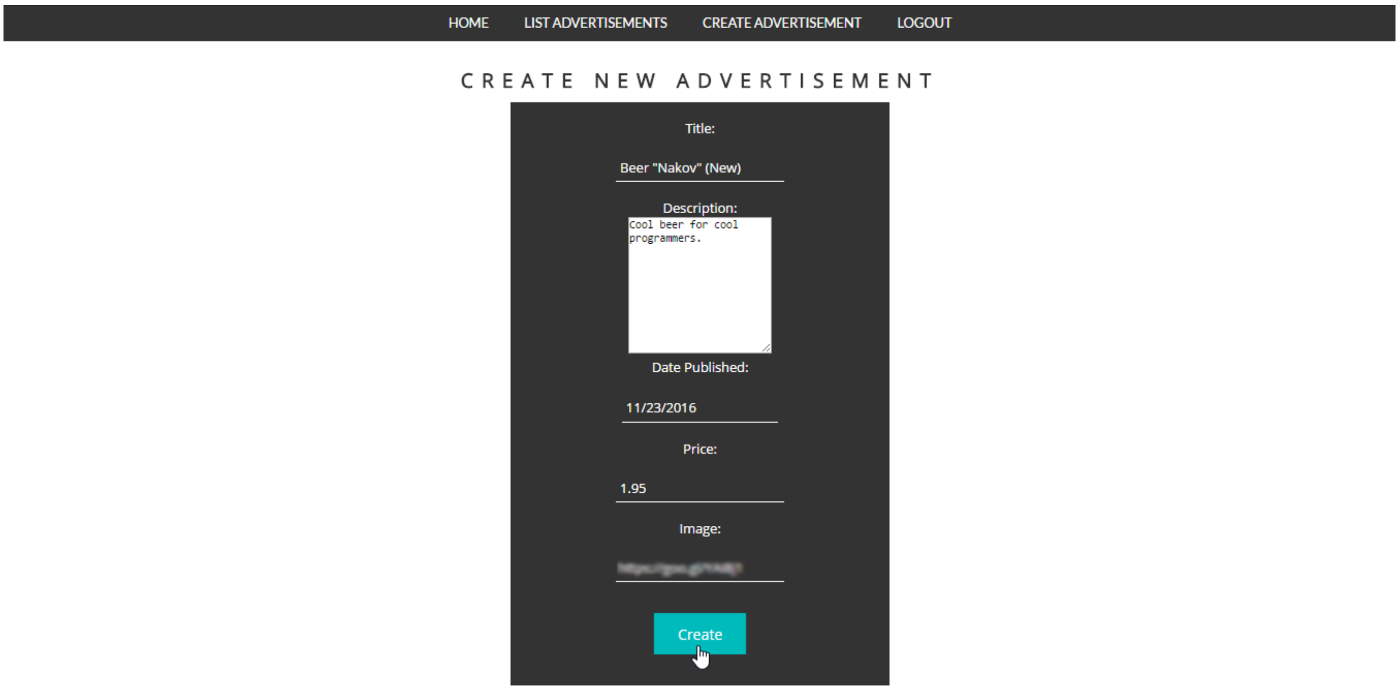
What have to be done is:

1. Creating **one more view**("section"), which will be used to **display** one **single advertisement**. The view should be accessible when you list all advertisements (by creating another **link** – "**Read more**" e.g.).
2. **Add** another **property** to the advertisements which should be **integer** – the **count** of **views** for every advert. Every advert should have **0** views **initially**. And that count will **grow** when someone tries to **display** its **single** view (when the "Read more" link is clicked).
3. Add also another **string** property containing **link** to some **image** (image will be used to display more information about current advert).
4. Whenever you list all the adverts, **sort** them **by views** count descending.

For more clarity check the screenshots below.

Screenshots

We should extend the create advert functionality. Now we should have **image** field:



CREATE NEW ADVERTISEMENT

Title:

Beer "Nakov" (New)

Description:

Cool beer for cool programmers.

Date Published:

11/23/2016

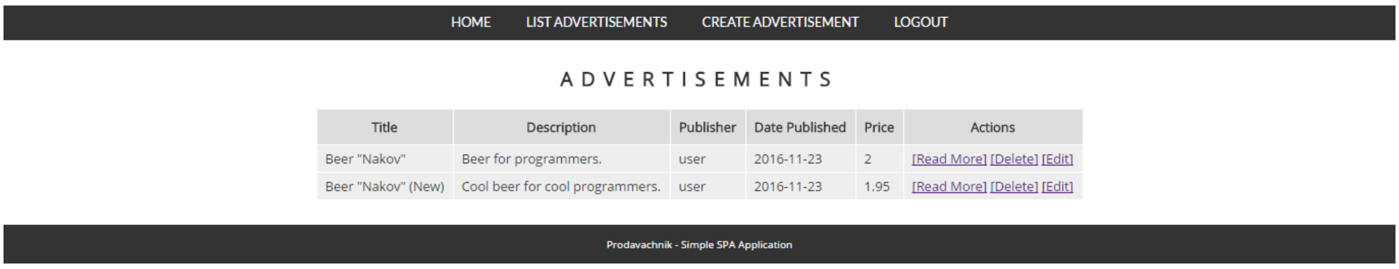
Price:

1.95

Image:

[redacted]

And when all adverts are listed: new “Read more” link:

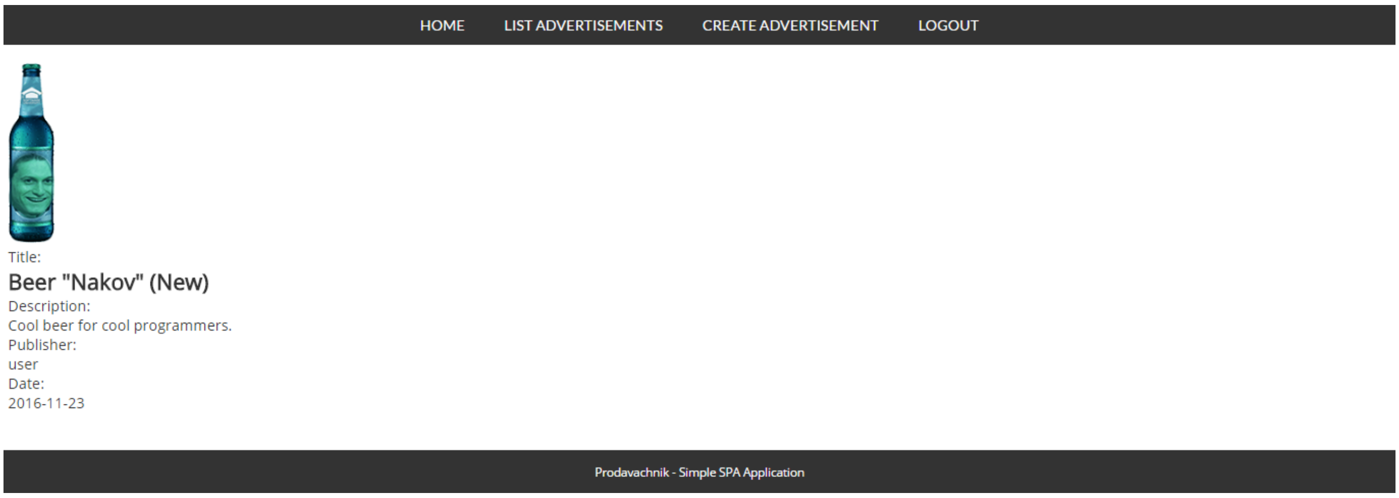


ADVERTISEMENTS

Title	Description	Publisher	Date Published	Price	Actions
Beer "Nakov"	Beer for programmers.	user	2016-11-23	2	[Read More] [Delete] [Edit]
Beer "Nakov" (New)	Cool beer for cool programmers.	user	2016-11-23	1.95	[Read More] [Delete] [Edit]

Prodavachnik - Simple SPA Application

Cool. Let’s check the “**Read more**” out:



Title:

Beer "Nakov" (New)

Description:

Cool beer for cool programmers.

Publisher:

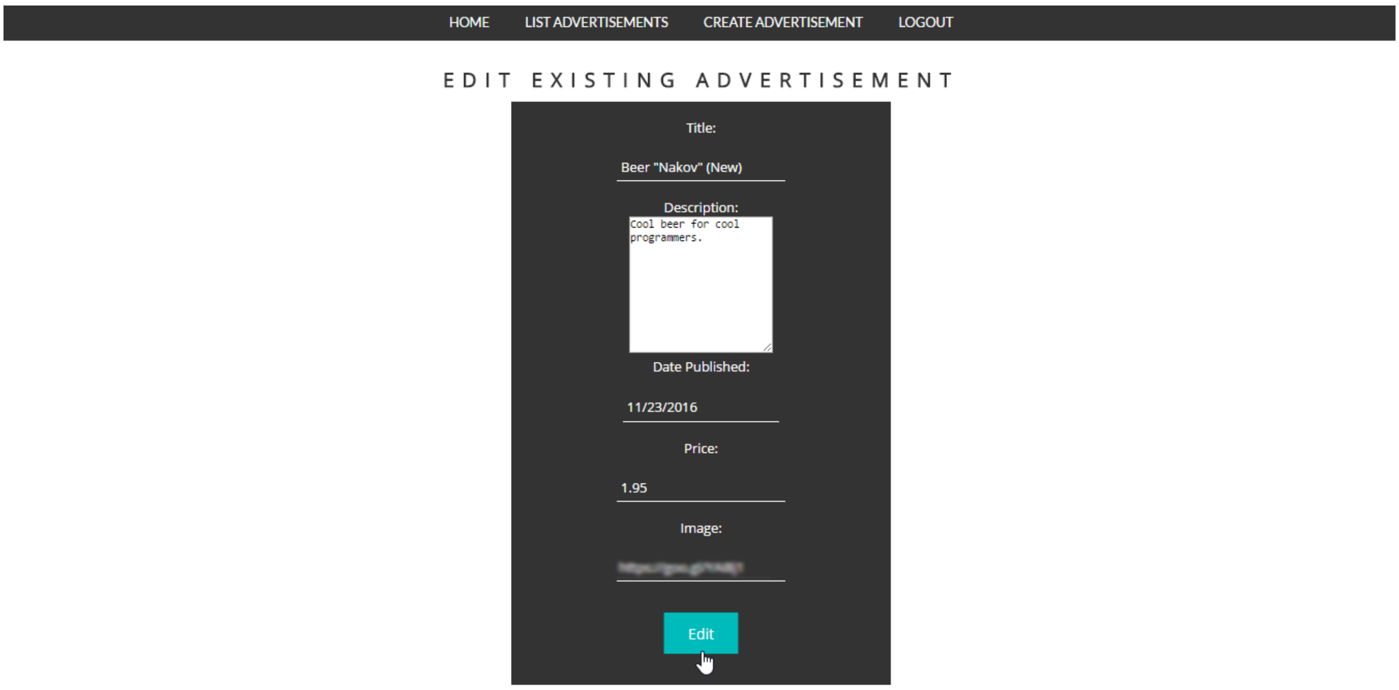
user

Date:

2016-11-23

Prodavachnik - Simple SPA Application

Here we have our **detailed view** about particular advert. What we have in addition to the previous one is that we have a **photo** and a **views** counter.



We should be able to **change** the **image** later on (if we want to):

Hints

- For the details view you can just add another section and give it some id, so you can later on refer it.
- Then on the part where **listing** of adverts is done, just add another action(<a> tag) – next to **delete** and **edit**: **Create** the "Read more" **link** and attach action:

  ```
  let readMoreLink = $('<a href="#">[Read More]</a>')
      .click(function() { displayAdvert(advert) });
  ```

  Add it to all links:

  ```
  links = [readMoreLink, ' ', deleteLink, ' ', editLink];
  ```

- The action should **render** some **html** – nothing special: some html tags with some content in it. Here is an **example** of how it might look like:

  ```
  function displayAdvert(advert){
      $('#viewDetailsAd').empty();

      let advertInfo = $('<div>').append(
          $('<br>'),
          $('<label>').text('Title:'),
          $('<h1>').text(advert.title),
          $('<label>').text('Description:'),
          $('<p>').text(advert.description),
          $('<label>').text('Publisher:'),
          $('<div>').text(advert.publisher),
          $('<label>').text('Date:'),
          $('<div>').text(advert.datePublished));

      $('#viewDetailsAd').append(advertInfo);
  ```

  ***Remember** later on when you add the new properties to include them here as well.
- Editing an advert should now also change the **image** url for the advert.

Note that the variables' names might differ from the ones in your project.

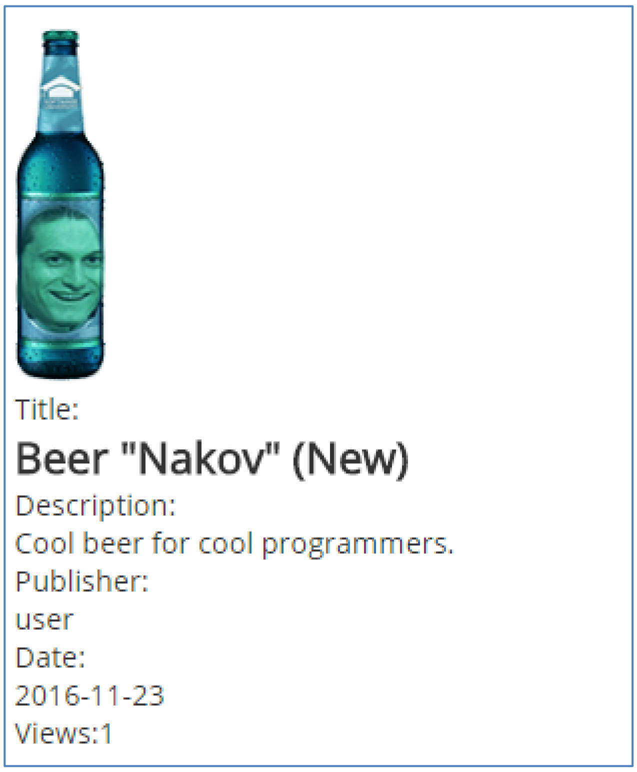
3. ***Add Viewcounter

Your task is to extend the functionality of the above project (when completed):

1. **Add** another **property** to the advertisements which should be **integer** – the **count** of **views** for every advert. Every advert should have **0** views **initially**. And that count will **grow** when someone tries to **display** its **single** view (when the "Read more" link is clicked).
2. Whenever you list all the adverts, **sort** them by **views count,** descending.

For more clarity check the screenshots below.

Screenshots



Title:
Beer "Nakov" (New)
Description:
Cool beer for cool programmers.
Publisher:
user
Date:
2016-11-23
Views:1

Hints

Implementing this functionality requires writing some business logic for your Kinvey app – since only the person who created an ad can modify it in the database, you can't just keep the views in the same object and update it when requested. Research business logic and collection access in Kinvey docs.