Andrew Pierno
Assignment III Part II
CS 496_400

The url structure is split into two different main routes, one for users, and one for products. The users route is structured as follows:

'get /3/users'
'get /3/users/:id'
'post /3/users'
'put /3/users'
'delete /3/users/:id'

The first get is a request to get all users. The second get is to get a single user by id. The post to /users is to create a new user. The put is to update the user and the delete is to delete a single user. Parts of my approach are restful, although I cannot say that the way I have implemented the api is completely restful. For example, I tried to use javascript naming conventions for attribute names for ease of use for the consumer of the api. The first name of the user is in the attribute firstName, this is the same for the user's last name. One way in which this api is not completely restful is in the delete of all users. Currently there is no way to batch delete users. This is by design. I thought it would be too powerful a function since each time a user is deleted, the products associated with that user are also deleted. For this small application, this one function would essentially wipe that database clean, which I did not want the consumer of the api to be able to do. Also, there is no way to really query the data at any level of granularity other than at the entire user level. You cannot, for example, get all users whose name starts with 'a', or who has more than 10 products in the 'store'.
My initial concept didnt have the users model at all so there were no users to model. Now however, users have a relationship with products where each product has exactly one user and each user can have zero to many products.

The products routes are structured as follows:

'get /3/products'
'get /3/products/:id'
'post /3/products'
'put /3/products'
'delete /3/products/:id'

The products routes are very similar to the users routes. we have a get all for /3/products, a get for an individual product at /products/:id, a post to create the product, a put to update the product and of course the ability to delete a product by id. Similar to the users routes, there is no way to batch delete say all the products for a certain user while still keeping the user profile. This again was by design but it should be noted because it makes this api not completely

RESTful. The naming convention again tried to mimic javascript naming conventions as suggested in ‘Web API Design’ by apigee.

In general, the app will never throw an error code other than a ‘200’ even if there is an error. I saw this pattern in the book and it is one that facebook utilizes. This is not to say that errors aren’t reported; in  my API all errors are in an errors array and is sent down with every request. If it is empty, there were no errors, otherwise the errors will be returned in the array. Also, the book mentioned the concept of versioning your APIs so that consumers can use the API knowing it will not change on them. I did not version this API. I put it under the root /3 for assignment 3 as I am maintaining assignments 1 and 2 on the same server but a better idea ( and what I would try next time) is to version the api so that as things change or the app expands, the consumers will be unaffected, and when ready, the api can be deprecated in favor of a newer API version.

In terms of changes to the schema from last week, I added the ability to associate users with products which gives a little clearer picture of what this app may eventually be used for, namely a store where users can create products for sale. In the future I anticipate adding the ability to upload images to products and giving each user a role ( of ‘buyer’ or ‘seller’ ) and then allowing ‘buyers’ to purchase items ( through some kind of payment processor integration ) and the ‘sellers’ to collect the money from purchased items. Ideally this site would be similar to something like etsy.com.

If I had to do this again, I would try to make the API 100% restful ( or as close as I could ) just to see what that actually was like to do. I think it is a little impractical at times but it’s nice to learn the rules before you break them. Also, knowing that there would be different user roles in the future, I would have liked to add some extra attributes to the user model so that the user model could support roles and have attributes that the ‘buyers’ of products will need to have in the future. This way I could have had a nearly completed users model before building the next segment of the app.

Below are the results from my curl tests. They passed at a rate of 100% and show the responses from the server. As you can see there is an error array in each of the responses that is empty. If there were attributes that failed validation, or any other error occurred during the request. They would be placed in the error array.  One note, is that if you are going to use the curl requests as is, you need to make sure to change the hard coded ID’s. The ID’s in here are ones that were used during development and may have been deleted.

Curl Tests and responses:
 - GET ALL USERS : curl http://52.34.84.150/3/users


```
{
 "users": [
   {
```

```
      "firstName": "testd",
      "lastName": "tesddt",
      "email": "emaweril@gmail.comd",
      "password": "laksdjf",
      "createdAt": "2016-01-24T00:34:29.028Z",
      "updatedAt": "2016-01-24T00:34:29.028Z",
      "id": "56a41c15c2bea6f40ea2402b"
    },
    {
      "firstName": "testwd",
      "lastName": "tesdwdt",
      "email": "emawerilf@gmail.comd",
      "password": "laksdjf",
      "createdAt": "2016-01-24T00:34:34.145Z",
      "updatedAt": "2016-01-24T00:34:34.145Z",
      "id": "56a41c1ac2bea6f40ea2402c"
    }
  ],
  "error": []
}
```


- GET SINGLE USER: curl http://52.34.84.150/3/users/56a41c1ac2bea6f40ea2402c


```
{
 "user": {
   "firstName": "testd",
   "lastName": "tesddt",
   "email": "emaweril@gmail.comd",
   "password": "laksdjf",
   "createdAt": "2016-01-24T00:34:29.028Z",
   "updatedAt": "2016-01-24T00:34:29.028Z",
   "id": "56a41c15c2bea6f40ea2402b"
 },
 "error": []
}
```


- CREATE: curl -d "firstName=andrew&lastName=pierno&email=andrepierno@gmail.com&password=andrewpierno" http://52.34.84.150/users


```
{
  "user": {
    "firstName": "Andrew",
```

```
    "lastName": "Pierno",
    "email": "andrewpierno@gmail.com",
    "password": "andrewpierno",
    "createdAt": "2016-01-24T01:09:23.586Z",
    "updatedAt": "2016-01-24T01:09:23.586Z",
    "id": "56a424435cc95da51048106e"
  },
  "error": []
}
```


- UPDATE: curl -X PUT -H "Cache-Control: no-cache" -d "id=56a3f16affbcb1cb0a3adfa8&firstName=Not Andrew&lastName=pierno&email=andrepierno@gmail.com&password=andrewpierno" 'http://52.34.84.150/3/users'


```
{
  "user": [
    {
      "firstName": "Not Andrew",
      "lastName": "Pierno",
      "email": "andrewpierno@gmail.com",
      "password": "andrewpierno",
      "createdAt": "2016-01-24T01:09:23.586Z",
      "updatedAt": "2016-01-24T01:10:51.831Z",
      "id": "56a424435cc95da51048106e"
    }
  ],
  "error": []
}
```


- DELETE: curl -X DELETE -H "Cache-Control: no-cache" 'http://52.34.84.150/3/users/56a424435cc95da51048106e'


```
{
  "errors": [],
  "deletedUser": [
    {
      "firstName": "Not Andrew",
      "lastName": "Pierno",
      "email": "andrewpierno@gmail.com",
      "password": "andrewpierno",
      "createdAt": "2016-01-24T01:11:48.121Z",
```

```
      "updatedAt": "2016-01-24T01:11:48.121Z",
      "id": "56a424d48e1c23ca10d820cf"
    }
  ],
  "deletedProducts": [
    "56a424ef8e1c23ca10d820d0",
    "56a424f68e1c23ca10d820d1",
    "56a424f98e1c23ca10d820d2"
  ]
}
```


Products Test

- GET ALL PRODUCTS (returns the user that the product is associated with as well) : curl http://52.34.84.150/3/products


```
{
 "products": [
  {
    "user": {
      "firstName": "Not Andrew",
      "lastName": "Pierno",
      "email": "andrewpierno@gmail.com",
      "password": "andrewpierno",
      "createdAt": "2016-01-24T01:11:48.121Z",
      "updatedAt": "2016-01-24T01:11:48.121Z",
      "id": "56a424d48e1c23ca10d820cf"
    },
    "name": "1",
    "cost": 4.4,
    "quantity": 7,
    "startDate": "1970-01-01T00:00:00.000Z",
    "color": "#eb6a5a",
    "gender": "female",
    "season": [
      "spring"
    ],
    "createdAt": "2016-01-24T01:12:15.329Z",
    "updatedAt": "2016-01-24T01:12:15.329Z",
```

```
  "id": "56a424ef8e1c23ca10d820d0"
},
{
  "user": {
    "firstName": "Not Andrew",
    "lastName": "Pierno",
    "email": "andrewpierno@gmail.com",
    "password": "andrewpierno",
    "createdAt": "2016-01-24T01:11:48.121Z",
    "updatedAt": "2016-01-24T01:11:48.121Z",
    "id": "56a424d48e1c23ca10d820cf"
  },
  "name": "2",
  "cost": 4.4,
  "quantity": 7,
  "startDate": "1970-01-01T00:00:00.000Z",
  "color": "#eb6a5a",
  "gender": "female",
  "season": [
    "spring"
  ],
  "createdAt": "2016-01-24T01:12:22.658Z",
  "updatedAt": "2016-01-24T01:12:22.658Z",
  "id": "56a424f68e1c23ca10d820d1"
},
{
  "user": {
    "firstName": "Not Andrew",
    "lastName": "Pierno",
    "email": "andrewpierno@gmail.com",
    "password": "andrewpierno",
    "createdAt": "2016-01-24T01:11:48.121Z",
    "updatedAt": "2016-01-24T01:11:48.121Z",
    "id": "56a424d48e1c23ca10d820cf"
  },
  "name": "3",
  "cost": 4.4,
  "quantity": 7,
  "startDate": "1970-01-01T00:00:00.000Z",
  "color": "#eb6a5a",
  "gender": "female",
  "season": [
    "spring"
```

```
      ],
      "createdAt": "2016-01-24T01:12:25.009Z",
      "updatedAt": "2016-01-24T01:12:25.009Z",
      "id": "56a424f98e1c23ca10d820d2"
    }
  ]
}
```


- GET SINGLE PRODUCT: curl http://52.34.84.150/3/products/5693c551ca83bd1d044ea68f


```
 {
 "product": {
   "user": {
     "firstName": "Not Andrew",
     "lastName": "Pierno",
     "email": "andrewpierno@gmail.com",
     "password": "andrewpierno",
     "createdAt": "2016-01-24T01:11:48.121Z",
     "updatedAt": "2016-01-24T01:11:48.121Z",
     "id": "56a424d48e1c23ca10d820cf"
   },
   "name": "3",
   "cost": 4.4,
   "quantity": 7,
   "startDate": "1970-01-01T00:00:00.000Z",
   "color": "#eb6a5a",
   "gender": "female",
   "season": [
     "spring"
   ],
   "createdAt": "2016-01-24T01:12:25.009Z",
   "updatedAt": "2016-01-24T01:12:25.009Z",
   "id": "56a424f98e1c23ca10d820d2"
 },
 "error": []
}
```


- CREATE: curl -d "name=4&user=56a424d48e1c23ca10d820cf&cost=5.5&quantity=5&gender=male&season=winter&color=#eb6a5a" http://52.34.84.150/products


```
 {
```

```
  "product": {
    "name": "4",
    "cost": 5.5,
    "quantity": 5,
    "startDate": "1970-01-01T00:00:00.000Z",
    "color": "#eb6a5a",
    "gender": "male",
    "season": [
      "winter"
    ],
    "user": "56a424d48e1c23ca10d820cf",
    "createdAt": "2016-01-24T01:13:54.385Z",
    "updatedAt": "2016-01-24T01:13:54.385Z",
    "id": "56a42552068b77df107bee44"
  },
  "error": []
}
```


- UPDATE:  curl -X PUT -H "Cache-Control: no-cache"  -d "name=4&user=56a424d48e1c23ca10d820cf&cost=5.5&quantity=5&gender=male&season=winter&color=#eb6a5a&id=56a42552068b77df107bee44" 'http://52.34.84.150/3/products'


```
  {
    "product": [
      {
        "name": "4",
        "cost": 5.5,
        "quantity": 5,
        "startDate": "1970-01-01T00:00:00.000Z",
        "color": "#eb6a5a",
        "gender": "male",
        "season": [
          "winter"
        ],
        "user": "56a424d48e1c23ca10d820cf",
        "createdAt": "2016-01-24T01:13:54.385Z",
        "updatedAt": "2016-01-24T01:15:13.543Z",
        "id": "56a42552068b77df107bee44"
      }
    ],
    "error": []
  }
```

- DELETE: curl -X DELETE -H "Cache-Control: no-cache" 'http://52.34.84.150/3/products/56a42552068b77df107bee44'


```
{
 "error": [],
 "deleted": [
  {
   "name": "4",
   "cost": 5.5,
   "quantity": 5,
   "startDate": "1970-01-01T00:00:00.000Z",
   "color": "#eb6a4a",
   "gender": "male",
   "season": [
    "winter"
   ],
   "user": "56a424d48e1c23ca10d820cf",
   "createdAt": "2016-01-24T01:13:54.385Z",
   "updatedAt": "2016-01-24T01:15:26.816Z",
   "id": "56a42552068b77df107bee44"
  }
 ]
}
```