CT30A9301 Code Camp on Platform Based Application Development

Lappeenranta 2015

Open Data Code Camp
Lecture notes report

Kalle Koponen, kalle.koponen@lut.fi, 0358454
Markus Salminen, markus.salminen@lut.fi, 0342408
Juri Pesonen, juri.pesonen@lut.fi, 0342327

Idea

The theme for the code camp was Open Data / Green IT. With those in mind we started to think for a program idea that could serve as a some kind of open API, but would also have a frontend implementation. We searched for interesting open API's to catch our interest, but didn't have any particular one in mind. First we thought we could implement the idea of open data by fetching data from open API and use it some way to have a implementation of our own. Soon we noticed that there were so many different kind of implementations that use for example Twitter feeds to analyze things. We tried to think outside of the box, but every time we came up with an idea, we noticed that it was already done by someone. We had some ideas like use IBM Watson image analyzer to analyze instagram posts to find out what was posted on the internet on that moment. That was done, not by using Watson, but inside instagram itself.

After trashing most of our ideas by just using google and finding out that someone has done our thing, we switched up the idea of how to use open data. We decided to gather our own data and use that as we please. We set up a question to help ourselves in brainstorming: "What kind of open API I would need right now?". Eventually we got the idea of open lecture notes database. To narrow it down, we decided to include only courses from Lappeenranta University of Technology. There it was, the idea for open data code camp. We felt that with our experience in web development, just implementing the API and a user interface for it would not take the whole week of coding, so we decided to spice things up. We wanted to try out something new and fascinating technology. We have actively used google docs for writing group works in school and noticed how good and easy it is to collaborate with. We wanted to do something similar and thats how we came up with an idea to use socket.io for adding a collaborating feature to the implementation.

Later on in the project we felt we do not have enough work still for the whole week and we decided to add something new. We thought that because of the open API nature, we wanted to keep the notes editable by everyone and didn't want to implement any kind of restrictions for the users. Although we wanted to make sure notes that are posted to the database stays there even if some mean person wants to destroy them. So we came up another feature inspired by Google Docs: revision history.

Technologies

Node.js w/ express - server side programming
MongoDB - database
Angularjs - client side programming
Socket.IO - realtime communication between server and client
Bootstrap CSS - html styling

API endpoint

Method	URL	Usage
POST	/courses/<id>/notes	Creating new note
GET	/courses/<id>/notes/<id>	Get note details
GET	/courses/<id>/notes	List all notes by course
GET	/courses	List all courses
GET	/snapshots/<id>	List history snapshots for note
PUT	/courses/<id>/notes/<id>	Edit note by id

Implementation

We begun the implementation by listing some of the main features and making a quick draft of the API and user interface. Then we distributed the work between team members. After a quick test of Cloud9 platform we decided to move back to using local development environments. This was because our project team had previously worked together and had a existing and working workflow.

Our first tasks were to create the API, MongoDB data model, and simple web frontend user interface. This part of the project was quite simple and didn’t cause many problems. Time was mostly spent on setting up local development environments, learning Node.JS, MongoDB, and AngularJS. Each team member worked on their own part of the project while also helping each other when needed. Whenever the project specifications needed clarification, the team members available discussed about it face-to-face. Very little documentation was created during the project due to team always working together in a shared space. In general the team was self organizing. Members moved on to next tasks when previous one was finished and help was given when needed.

After each member had finished their part of the initial goal, the parts were integrated and any issues that came up were quickly solved. Late in the first day the first version was working and running. In this initial version the multiple person live editing didn’t yet work, but creation and editing of notes worked well. At this point we begun the work on live editing. For this we used Socket.IO, javascript library that uses HTML5 WebSockets.

Socket.IO made it very easy to get a message send and receive system to work. We were able to reuse large parts of the earlier API code. After this we ran into the two biggest issues in our project, caret behaviour in HTML textbox element and issues caused by asynchronous http requests. Caret would reset to the end of the textbox element after every update of the textbox content, and due to network lag when writing fast user agent would get updates on the note contents that were actually already outdated. The second issue was only noticed after we ran the code on a remote server. When running the program on local server none of us encountered any real issues with network lag because the latency was so low. However when the response time grew to 50-100ms, a fast writer would get reply messages containing data on current text of the lecture note which was actually already outdated. Our naive implementation then replaced the actual contents with this outdated data. On thursday we were working on fixing these issues, as well as improving the client design and creating a simple revision management for the lecture notes.

After some searching we found info on how to deal with issues of caret resetting itself by moving it back to its previous position after every update. A few minor tweaks were required so that when multiple users were editing the file their carets would stay at proper position when other people edited the text. Second issues of asynchronous requests and outdated data was only partially fixed on the friday morning, less than hour before deadline. After spending previous evening on it, we ended up with somewhat simple fix. We made the client simply ignore the responses for their own note update requests. However this didn’t fix issues that come up when multiple people edit the same document simultaneously. Final product works well when people edit it in turns, but more than one people writing at the exact same time causes issues with different clients overwriting each others versions.

Future

If we would continue to develop our project first we would fix the remaining lag issues when multiple people are using same note at the same time. The app works really well with one user but when multiple users are editing same note things start to go little wrong. We already have quite a lot of lag compensation but it is not perfect if latency is too high.

Another improvement would be to fetch and parse all the LUT courses from e.g. Uni. Currently the app has only a few courses. We were also thinking about creating some kind of search or filtering functionality for courses and notes.

Currently the app doesn’t really have a mobile layout. It does scale for different screen sizes and resolutions but the elements of the site stay side by side and don’t for example stack like the current trend is. Fully functional mobile layout would be a great feature.

Attachment 1. API GET response examples

GET /courses

Response body:

```
[
   {
      "name":"Games and Networking",
      "code":"CT30A5002",
      "_id":"5502b201f7d8951f36832785",
      "__v":0,
      "notes":[

      ]
   },
   {
      "name":"Wireless Service Engineering",
      "code":"CT30A8301",
      "_id":"5502b201f7d8951f36832788",
      "__v":0,
      "notes":[

      ]
   },
   {
      "name":"Parallel Computing",
      "code":"CT30A7500",
      "_id":"5502b201f7d8951f36832787",
      "__v":0,
      "notes":[

      ]
   },
   {
      "name":"Service Oriented Architecture",
      "code":"CT30A8902",
      "_id":"5502b201f7d8951f36832789",
      "__v":0,
      "notes":[

      ]
   },
   {
      "name":"Software Engineering Methods",
      "code":"CT60A5100",
      "_id":"5502b201f7d8951f3683278a",
      "__v":0,
      "notes":[

      ]
   },
   {
      "name":"Hajautetut järjestelmät",
      "code":"CT30A3400",
      "_id":"5502b201f7d8951f36832786",
      "__v":0,
      "notes":[

      ]
   },
   {
      "__v":1,
      "_id":"5502b201f7d8951f3683278b",
      "code":"CT60A7000",
      "name":"Critical Thinking and Argumentation in Software Engineering",
      "notes":[
         {
            "title":"Lecture 1",
            "note":"- Thinking is important",
            "created_at":"2015-03-13T09:46:56.061Z",
            "modified_at":"2015-03-13T09:47:33.387Z",
            "_id":"5502b2104d27055e360616e5"
         }
      ]
   }
]
```

GET /courses/<id>/notes

Response body:

```
[
  {
    "title": "Lecture 1",
    "note": "- Thinking is important\n\n- Argumentation is very important",
    "created_at": "2015-03-13T09:46:56.061Z",
    "modified_at": "2015-03-13T09:58:21.160Z",
    "_id": "5502b2104d27055e360616e5"
  },
  {
    "title": "Lecture 2",
    "note": "- Didn't go... was sick",
    "created_at": "2015-03-13T09:58:22.766Z",
    "modified_at": "2015-03-13T09:58:39.869Z",
    "_id": "5502b4be4d27055e360616ea"
  }
]
```

GET /courses/<id>/notes/<id>

Response body:

```
{
  "title": "Lecture 1",
  "note": "- Thinking is important\n\n- Argumentation is very important",
  "created_at": "2015-03-13T09:46:56.061Z",
  "modified_at": "2015-03-13T09:58:21.160Z",
  "_id": "5502b2104d27055e360616e5"
}
```

GET /snapshots/<id>

Response body:

```
[
  {
    "noteId": "5502b2104d27055e360616e5",
    "title": "L",
    "note": "",
    "created_at": "2015-03-13T09:46:58.016Z",
    "_id": "5502b2124d27055e360616e6",
    "__v": 0
  },
  {
    "noteId": "5502b2104d27055e360616e5",
    "title": "Lecture 1",
    "note": "- Thinking is importar",
    "created_at": "2015-03-13T09:47:31.357Z",
    "_id": "5502b2334d27055e360616e7",
    "__v": 0
  },
  {
    "noteId": "5502b2104d27055e360616e5",
    "title": "Lecture 1",
    "note": "- Thinking is important",
    "created_at": "2015-03-13T09:54:16.738Z",
    "_id": "5502b3c84d27055e360616e8",
    "__v": 0
  },
  {
    "noteId": "5502b2104d27055e360616e5",
    "title": "Lecture 1",
    "note": "- Thinking is important\n\n-",
    "created_at": "2015-03-13T09:58:05.189Z",
    "_id": "5502b4ad4d27055e360616e9",
    "__v": 0
  },
  {
    "noteId": "5502b2104d27055e360616e5",
    "title": "Lecture 1",
    "note": "B",
    "created_at": "2015-03-13T10:02:17.056Z",
    "_id": "5502b5a94d27055e360616ec",
    "__v": 0
  },
  {
    "noteId": "5502b2104d27055e360616e5",
    "title": "Lecture 1",
    "note": "- Big data\n\n- Not why but what\n- More\n- Messy\n- C",
    "created_at": "2015-03-13T10:02:48.456Z",
    "_id": "5502b5c84d27055e360616ed",
    "__v": 0
  },
  {
    "noteId": "5502b2104d27055e360616e5",
    "title": "Lecture 1",
    "note": "- Big data\n\n- Not why but what\n- More\n- Messy\n- C",
    "created_at": "2015-03-13T10:02:48.458Z",
    "_id": "5502b5c84d27055e360616ee",
    "__v": 0
  },
  {
    "noteId": "5502b2104d27055e360616e5",
    "title": "Lecture 1",
    "note": "- Big data\n\n- Not why but what\n- More\n- Messy\n- Correlation\n- Datafication\n-
Value\n- Implication\n- Risks\n- Control",
    "created_at": "2015-03-13T10:03:18.820Z",
    "_id": "5502b5e64d27055e360616ef",
    "__v": 0
  }
]
```

Attachment 2. POST & PUT request examples

`POST /courses/<id>/notes`

`Request body`

`Response body`

`PUT /courses/<id>/notes/<id>`

`Request body`

`Response body`