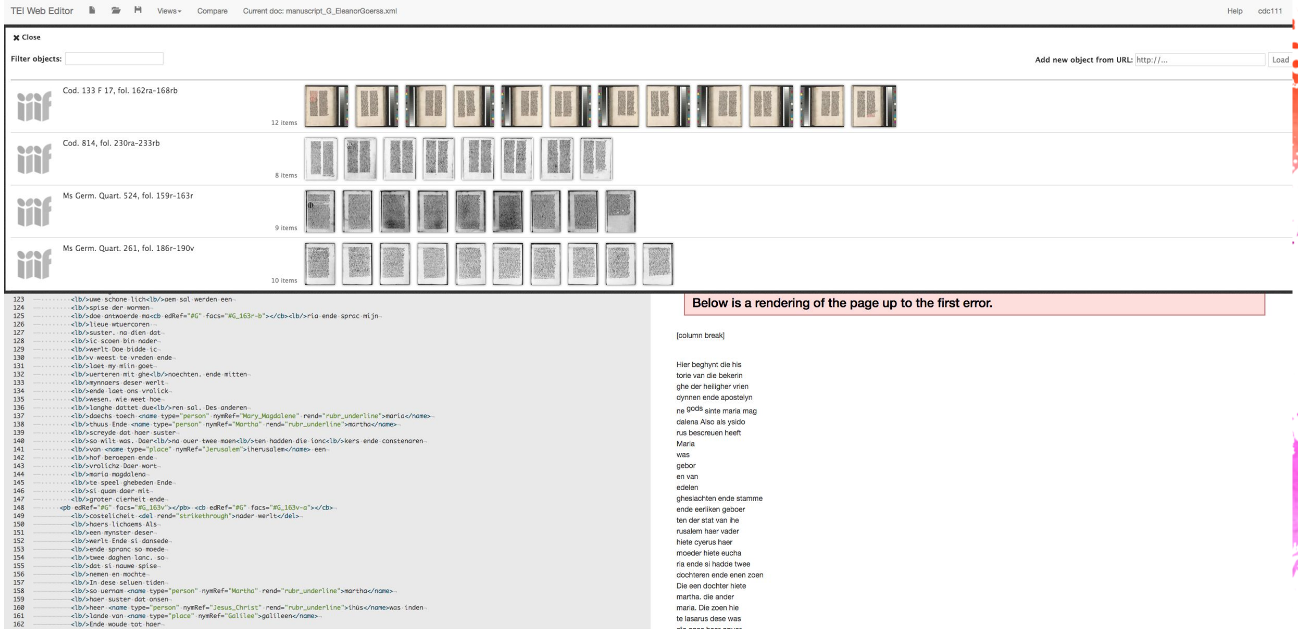
Building APIs

- When does your project need an API?
  - Rapidly changing data
  - Large volume of requests
  - Many people find it useful and want to build on top of it or extend it
  - *You* want to build on top of your data
  - Trust issues

Planning

- Stable data structure
- Think about your audiences: users and developers
  - How will they use it? (Features and requirements - what operations?)
  - What should be exposed?
  - How will users authenticate?
- Infrastructure: where will it live?
- Best Practices Article: https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api

Example: Web Editor



Current doc: manuscript_G_EleanorGoerss.xml

Filter objects:

Add new object from URL: http://...

Cod. 133 F 17, fol. 162ra-168rb — 12 items

Cod. 814, fol. 230ra-233rb — 8 items

Ms Germ. Quart. 524, fol. 159r-163r — 9 items

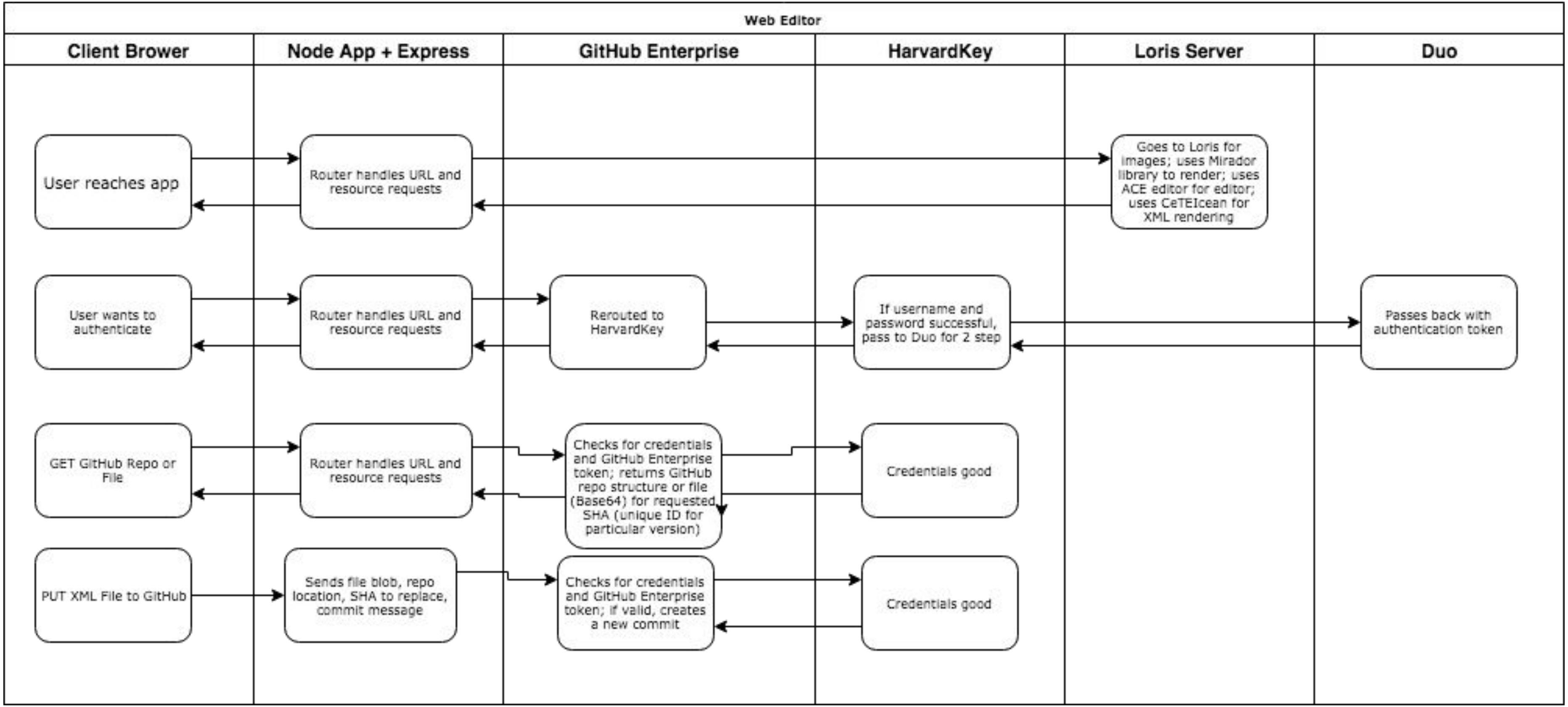
Ms Germ. Quart. 261, fol. 186r-190v — 10 items

```
            <lb/>uwe schone lich<lb/>aem sal werden een
            <lb/>spise der wormen
            <lb/>doe antwoerde ma<cb edRef="#G" facs="#G_163r-b"></cb><lb/>ria ende sprac mijn
            <lb/>lieue wtuercoren
            <lb/>suster. na dien dat
            <lb/>ic scoen bin nader
            <lb/>werlt Doe bidde ic
            <lb/>v weest te vreden ende
            <lb/>laet my miin goet
            <lb/>uerteren mit ghe<lb/>noechten. ende mitten
            <lb/>mynnaers deser werlt
            <lb/>ende laet ons vrolick
            <lb/>wesen. wie weet hoe
            <lb/>langhe dattet due<lb/>ren sal. Des anderen
            <lb/>daechs toech <name type="person" nymRef="Mary_Magdalene" rend="rubr_underline">maria</name>
            <lb/>thuus Ende <name type="person" nymRef="Martha" rend="rubr_underline">martha</name>
            <lb/>screyde dat haer suster
            <lb/>so wilt was. Daer<lb/>na ouer twee maen<lb/>ten hadden die ionc<lb/>kers ende constenaren
            <lb/>van <name type="place" nymRef="Jerusalem">iherusalem</name> een
            <lb/>hof beroepen ende
            <lb/>vrolichz Daer wort
            <lb/>maria magdalena
            <lb/>te speel ghebeden Ende
            <lb/>si quam daer mit
            <lb/>groter cierheit ende
         <pb edRef="#G" facs="#G_163v"></pb> <cb edRef="#G" facs="#G_163v-a"></cb>
            <lb/>costelicheit <del rend="strikethrough">nader werlt</del>
            <lb/>haers lichaems Als
            <lb/>een mynster deser
            <lb/>werlt Ende si dansede
            <lb/>ende spranc so moede
            <lb/>twee daghen lanc. so
            <lb/>dat si nauwe spise
            <lb/>nemen en mochte
            <lb/>In dese seluen tiden
            <lb/>so uernam <name type="person" nymRef="Martha" rend="rubr_underline">martha</name>
            <lb/>haer suster dat onsen
            <lb/>heer <name type="person" nymRef="Jesus_Christ" rend="rubr_underline">ihūs</name>was inden
            <lb/>lande van <name type="place" nymRef="Galilee">galileen</name>
            <lb/>Ende woude tot haer
```

Below is a rendering of the page up to the first error.

[column break]

Hier beghynt die his
torie van die bekerin
ghe der heiligher vrien
dynnen ende apostelyn
ne gods sinte maria mag
dalena Also als ysido
rus bescreuen heeft
Maria
was
gebor
en van
edelen
gheslachten ende stamme
ende eerliken geboer
ten der stat van ihe
rusalem haer vader
hiete cyerus haer
moeder hiete eucha
ria ende si hadde twee
dochteren ende enen zoen
Die een dochter hiete
martha. die ander
maria. Die zoen hie
te lasarus dese was



Web Editor

Client Brower

Node App + Express

GitHub Enterprise

HarvardKey

Loris Server

Duo

User reaches app

Router handles URL and resource requests

Goes to Loris for images; uses Mirador library to render; uses ACE editor for editor; uses CeTEIcean for XML rendering

User wants to authenticate

Router handles URL and resource requests

Rerouted to HarvardKey

If username and password successful, pass to Duo for 2 step

Passes back with authentication token

GET GitHub Repo or File

Router handles URL and resource requests

Checks for credentials and GitHub Enterprise token; returns GitHub repo structure or file (Base64) for requested SHA (unique ID for particular version)

Credentials good

PUT XML File to GitHub

Sends file blob, repo location, SHA to replace, commit message

Checks for credentials and GitHub Enterprise token; if valid, creates a new commit

Credentials good

Example: Harvard Art Museums

- Documentation: https://github.com/harvardartmuseums/api-doc
- Register: https://www.harvardartmuseums.org/collections/api
- API App Examples
  - http://apps.harvardartmuseums.org/art-explorer/
  - http://apps.harvardartmuseums.org/museum-explorer/
  - http://apps.harvardartmuseums.org/suns-explorer/

Example: Blog Application

NodeJS + Express

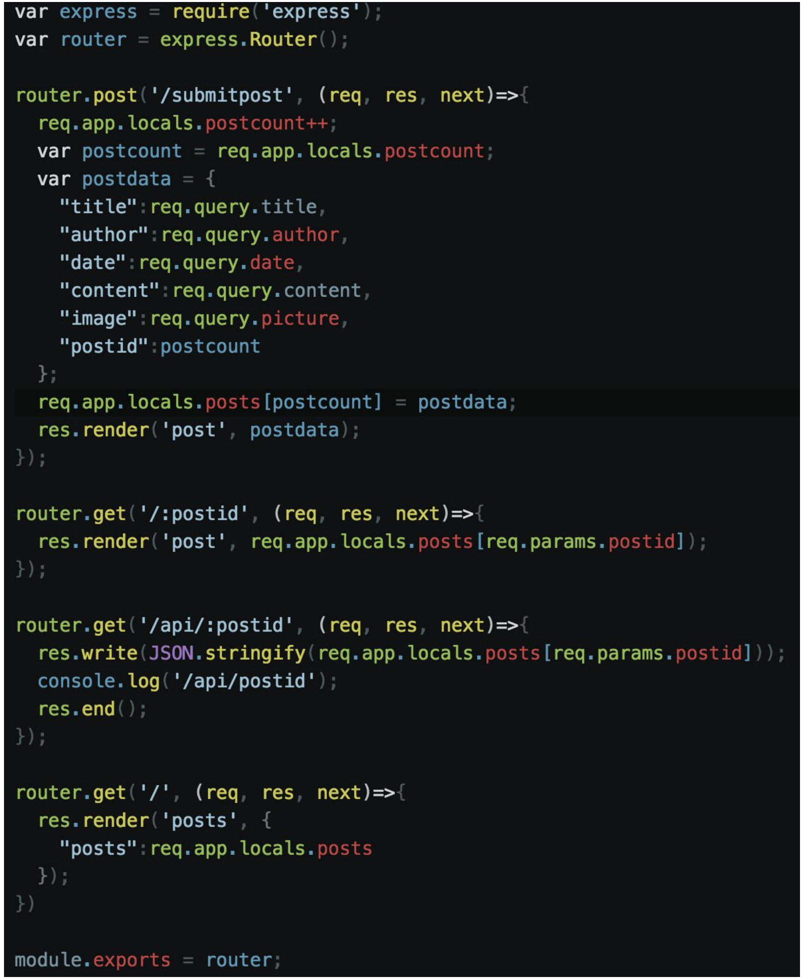
http://159.203.177.188:3000/posts

http://159.203.177.188:3000/posts/api/2

Python + Flask Resources

http://programminghistorian.github.io/ph-submissions/lessons/creating-apis-with-python-and-flask

https://auth0.com/blog/developing-restful-apis-with-python-and-flask/



```
var express = require('express');
var router = express.Router();

router.post('/submitpost', (req, res, next)=>{
  req.app.locals.postcount++;
  var postcount = req.app.locals.postcount;
  var postdata = {
    "title":req.query.title,
    "author":req.query.author,
    "date":req.query.date,
    "content":req.query.content,
    "image":req.query.picture,
    "postid":postcount
  };
  req.app.locals.posts[postcount] = postdata;
  res.render('post', postdata);
});

router.get('/:postid', (req, res, next)=>{
  res.render('post', req.app.locals.posts[req.params.postid]);
});

router.get('/api/:postid', (req, res, next)=>{
  res.write(JSON.stringify(req.app.locals.posts[req.params.postid]));
  console.log('/api/postid');
  res.end();
});

router.get('/', (req, res, next)=>{
  res.render('posts', {
    "posts":req.app.locals.posts
  });
})

module.exports = router;
```

CMS REST APIs

- Easiest way to add an API to your website
  - CMSes already force some structure on your content
- Omeka
  - http://omeka.readthedocs.io/en/latest/Reference/api/
- Scalar
  - http://scalar.usc.edu/works/guide2/working-with-the-api
- WordPress
  - Now in core: https://developer.wordpress.org/rest-api/
  - Example: http://darthcrimson.org/wp-json/wp/v2/posts
- Drupal
  - 7: https://www.drupal.org/project/restws
  - 8: REST is in core