# fruschtique

by Norbert Luttenberger

# Architecture

## Design and Implementation

# Guidelines

1. fruschtique provides a framework
   for displaying and analysing *collections* of recipes.
2. Collections are subdivided in *chapters*,
   every chapter holding several XML-coded *recipes*.
3. Each collection is held in a *git repository*; an additional
   single repository holds normalized ingredient names.
4. The fruschtique software system has a *3-tier architecture*,
   comprising interact, transform, and persist tiers.
5. Users interact via "static" web pages built with *html + css + js*,
   no application server required.

# 3-tier architecture

## interact

- users
  (public read access to transform results)
- authors
  (read/write access to repos)

## transform

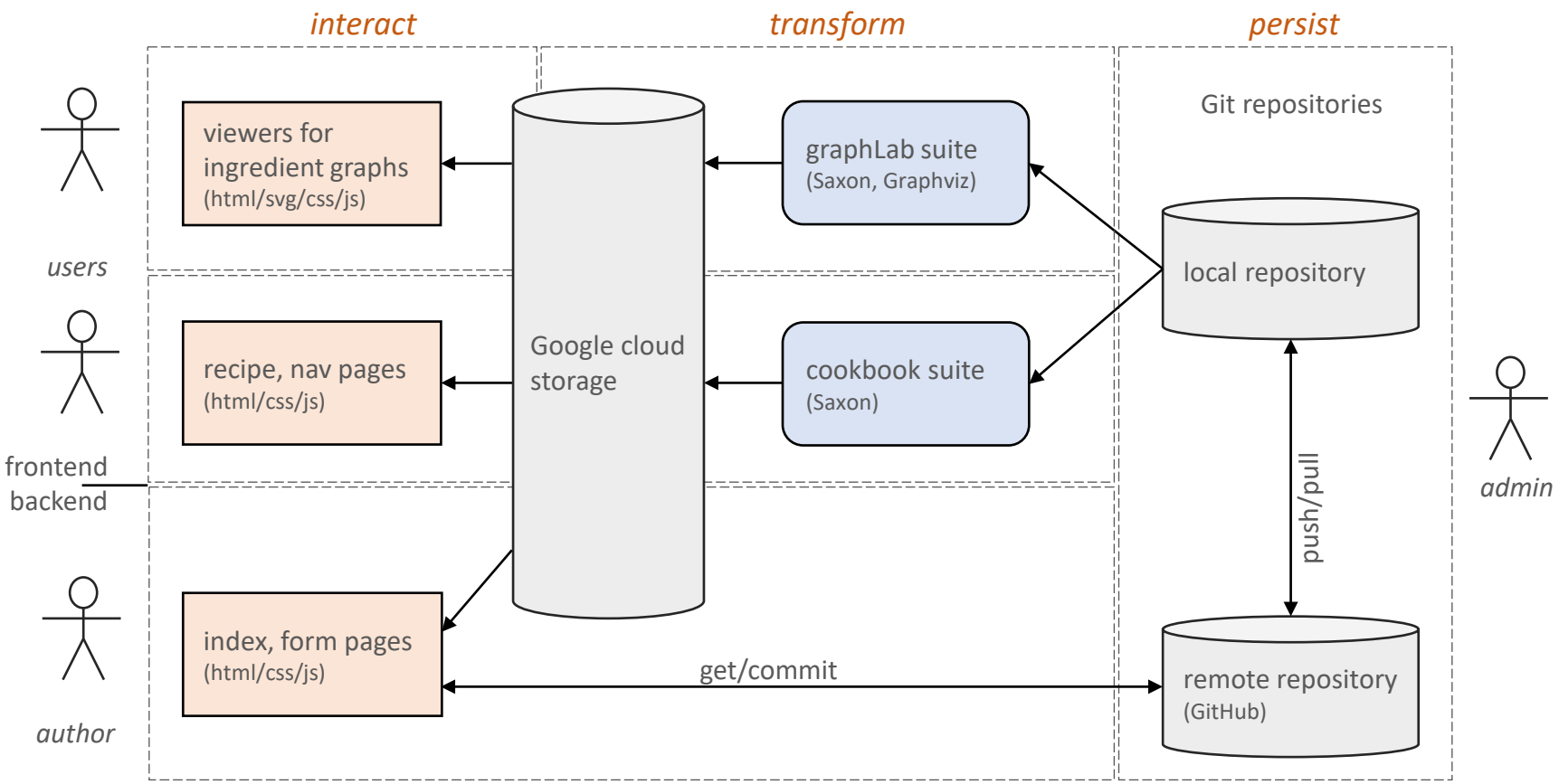
- cc repo*
  → cookbook pages
- cc repo*
  → graphLab viewer pages
- cc repo*
  → …

## persist

- Git repositories for culinary collections
  → *cc repos*
- XML coding for recipes
- Git repository for normalized ingredient names
  → *nn repo*

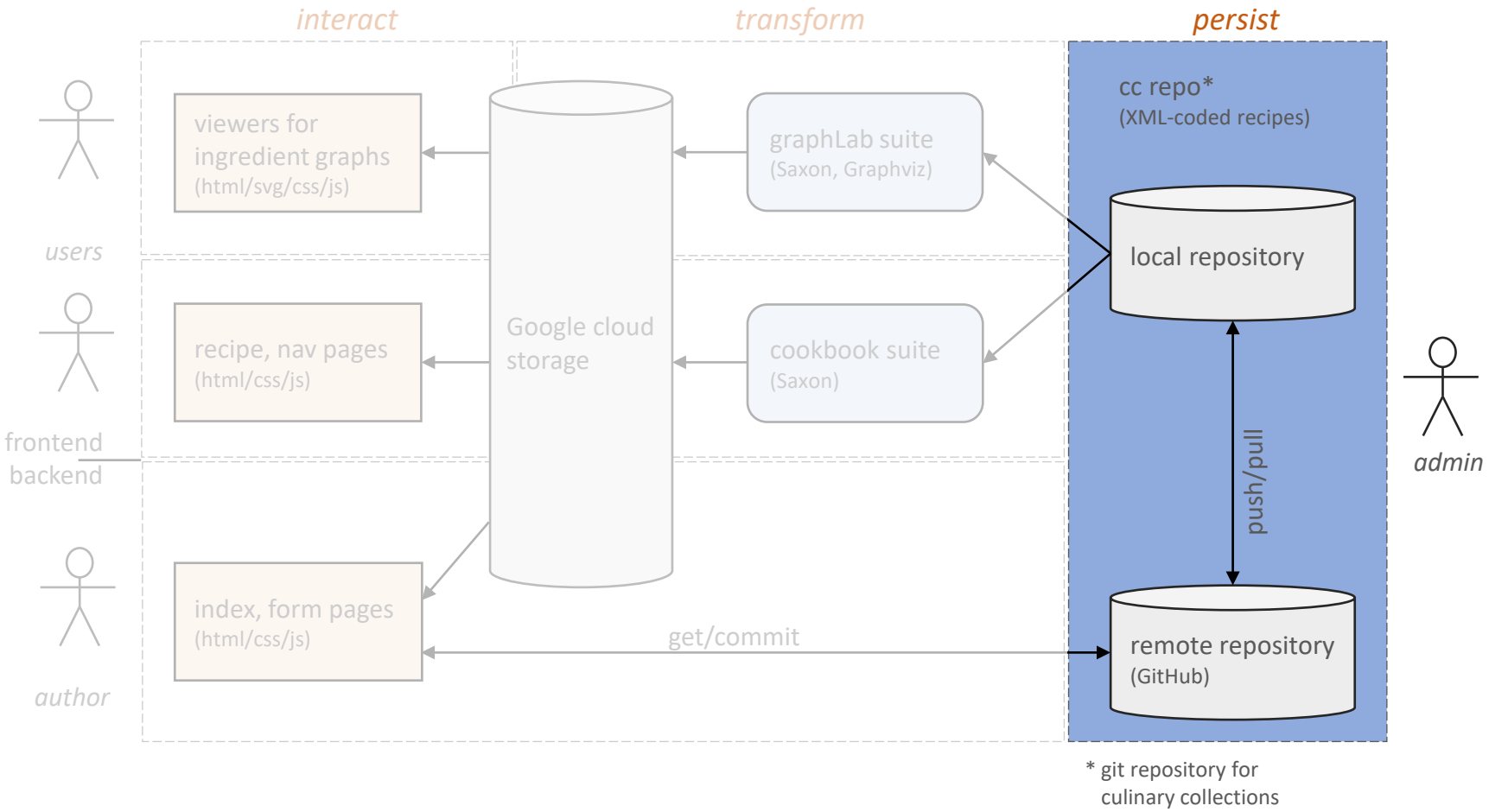
* repository for culinary collections

# 3-tier architecture




Persistant storage

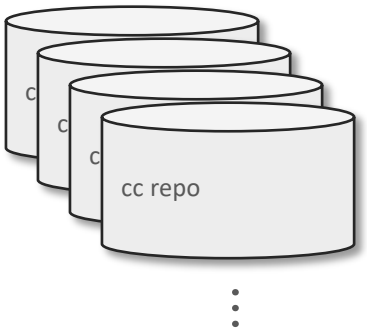
# 3-tier architecture



# Concept

- Use Git technology
  - → persistant storage
  - → versioning
- Pairs of local/remote repository
  - → enable author collaboration
- Remote repositories hosted by GitHub

# Repos for different purposes




culinary collection (cc) repos –
one repository per culinary collection

each repo composed of cc repo\local, and cc repo\remote


normalized ingredient names (nn) repo –
derived from *fruschtique controlled vocabulary for ingredient names* (frVocab)

→ Normalized ingredient names are needed for the construction of ingredient graphs

composed of nn repo\local, and nn repo\remote

# Culinary collection repos


- one cc repo per collection
- cc repo name = collection name

Inner structure:

- descriptor.json — additional info about collection
- recipes_xml — chapter/recipe hierarchy
  - 01 chap
    - recp a.xml
    - recp b.xml
    - …
  - 02 chap
  - …
- sublists_xml — named ingredients lists for export/import
  - aa.xml
  - bb.xml
  - …

# Normalized ingredient names repo


Inner structure:

- datalist-for-autocomplete.html
- igt-catalog.json

Both files are derived from the *fruschtique controlled vocabulary for ingredient names* (frVocab), see below.

# Create and fill Git repos (local/remote)

1. On GitHub: Create a new repository.
   To avoid errors, do **not** initialize the new repository with README, license, or gitignore files.
   You can add these files after your project has been pushed to GitHub.
2. On your local machine:
   Change the current working directory to your local project. Then open Git Bash and enter:

   ```
   $ git init -b main
   ```
   # Initializes the local directory as a Git repository, branch main.

   ```
   $ git add .
   ```
   #Adds the files in the local repository and stages them for commit.
   #To unstage a file, use 'git reset HEAD YOUR-FILE'.

   ```
   $ git commit -m "First commit"
   ```
   # Commits the tracked changes and prepares them to be pushed to a remote repository.
   #To remove this commit and modify the file, use 'git reset --soft HEAD~1' and commit and add the file again.
3. On GitHub:
   At the top of your repository's Quick Setup page, click to copy the remote repository URL.
4. On your local machine, in Git Bash:
   Add the URL for the remote repository where your local repository will be pushed.

   ```
   $ git remote add origin <remote repository URL>
   ```
   # Sets the new remote repository

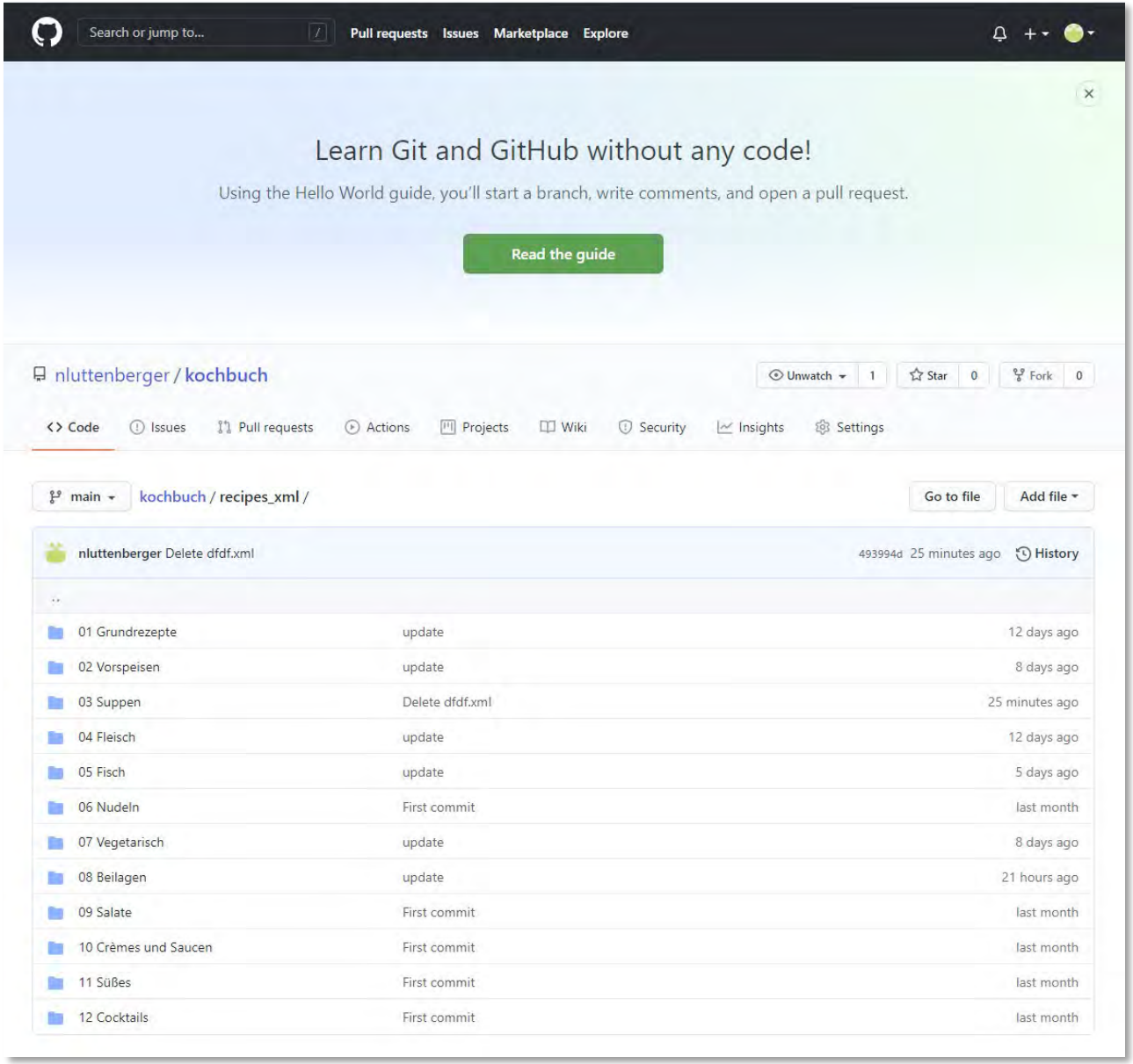
   ```
   $ git remote -v
   ```
   # Verifies the remote URL

   ```
   $ git push -u origin main
   ```
   # Pushes the changes in your local repository up to the remote repository you specified as the origin

https://docs.github.com/en/free-pro-team@latest/github/importing-your-projects-to-github/adding-an-existing-project-to-github-using-the-command-line

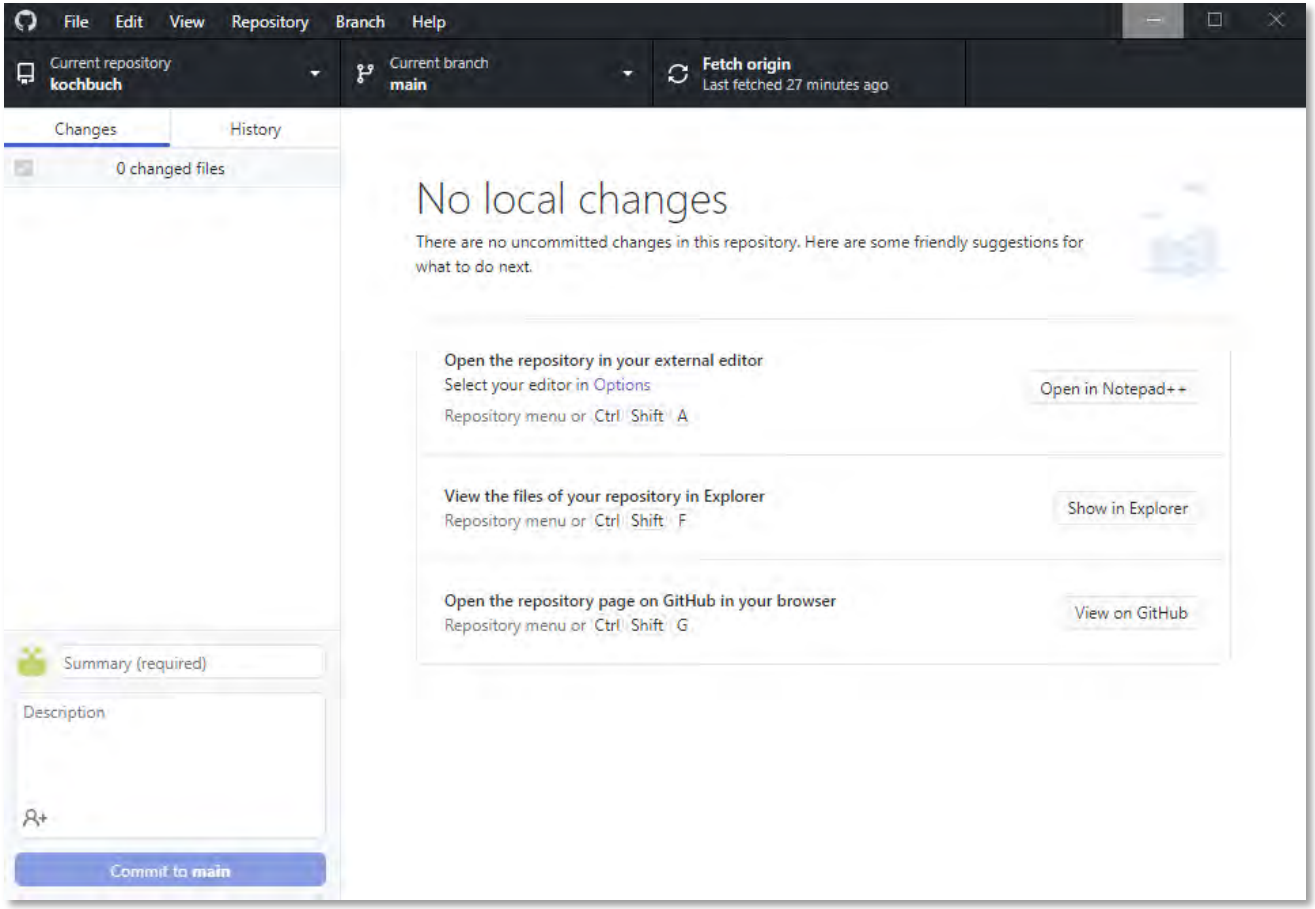
# Administration of cc repos

cc repo\remote:
GitHub web interface



# Administration of cc repos

cc repo\local:
GitHub desktop
(incl. push, pull)



# Access to repos

read and write access to repo\local

- graphLab suite
- cookbook suite

read access to repo\remote

- public

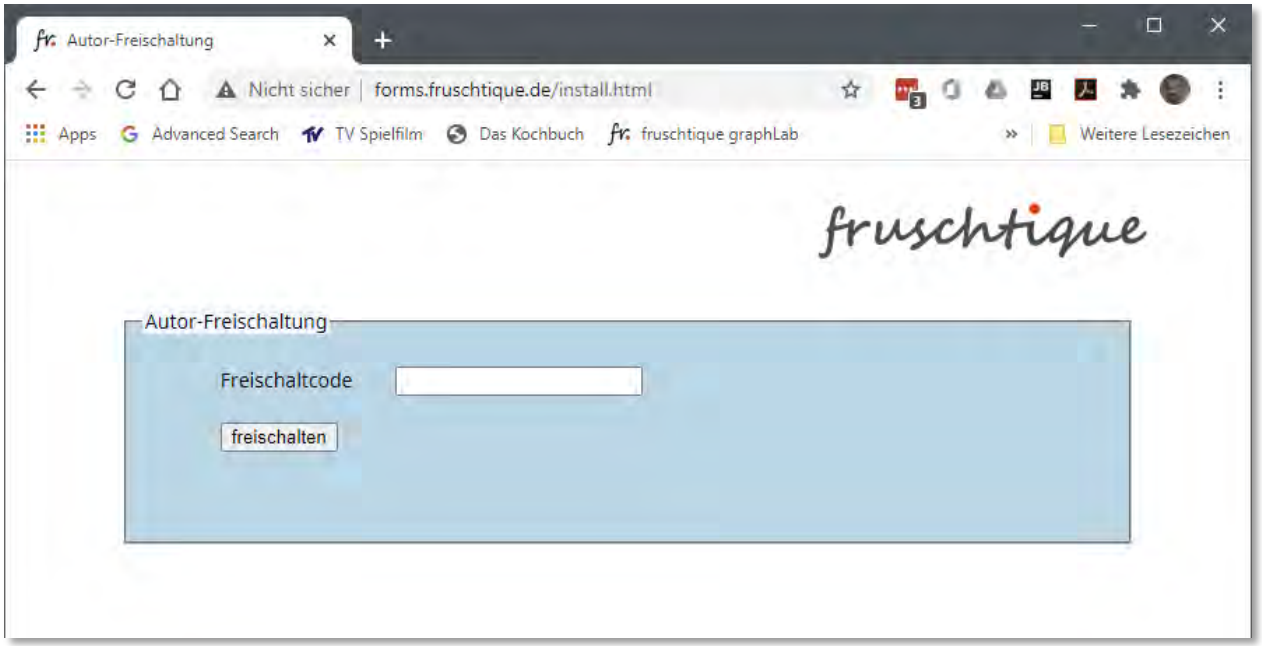
write access to repo\remote

- authors only
  via admin-distributed author key

# Write protection for cc repos\remote

- Let the admin forward an author key
  to designated authors on a secure path
- Author key is entered once into browser,
  i.e. never transmitted over the network
- Let the browser compute a GitHub
  Personal Access Token (PAT) from the author key
- When necessary, the PAT is transmitted to GitHub
  via https
- Let GitHub do the PAT checking

# Activate fruschtique authorship

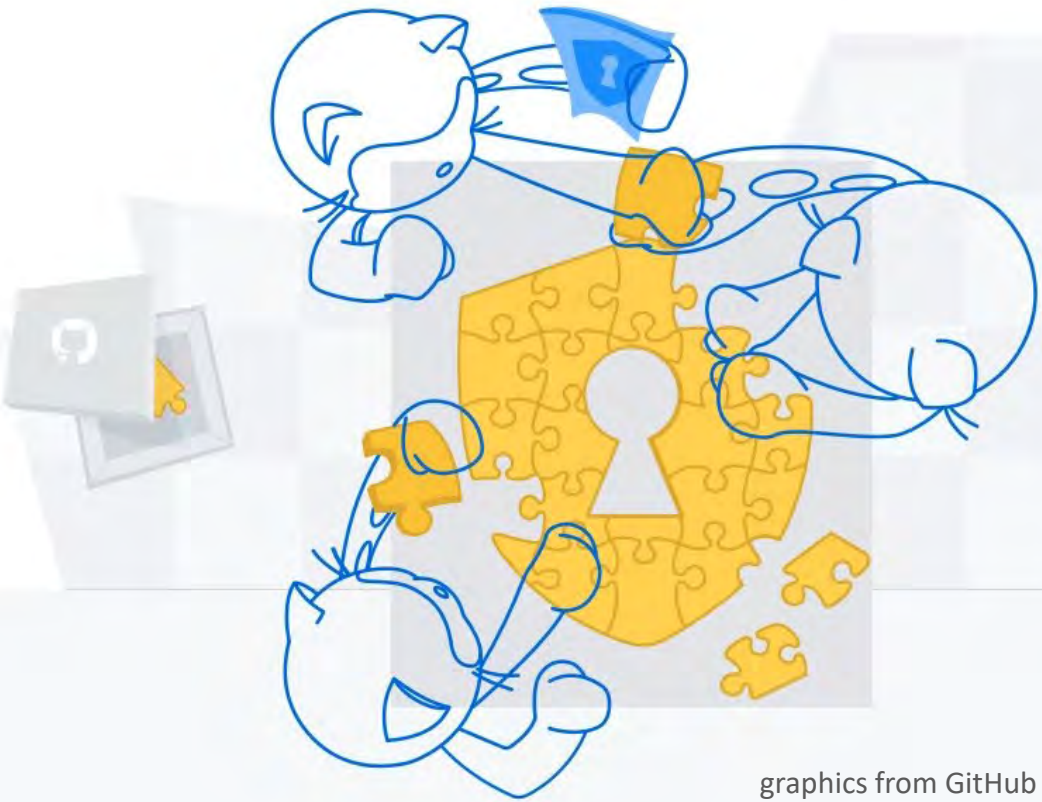



- get author key from admin
- call http://forms.fruschtique.de/install.html
- enter author key
- press "activate"

To be repeated for all browsers that the author wants to use with *http://forms.fruschtique.de*

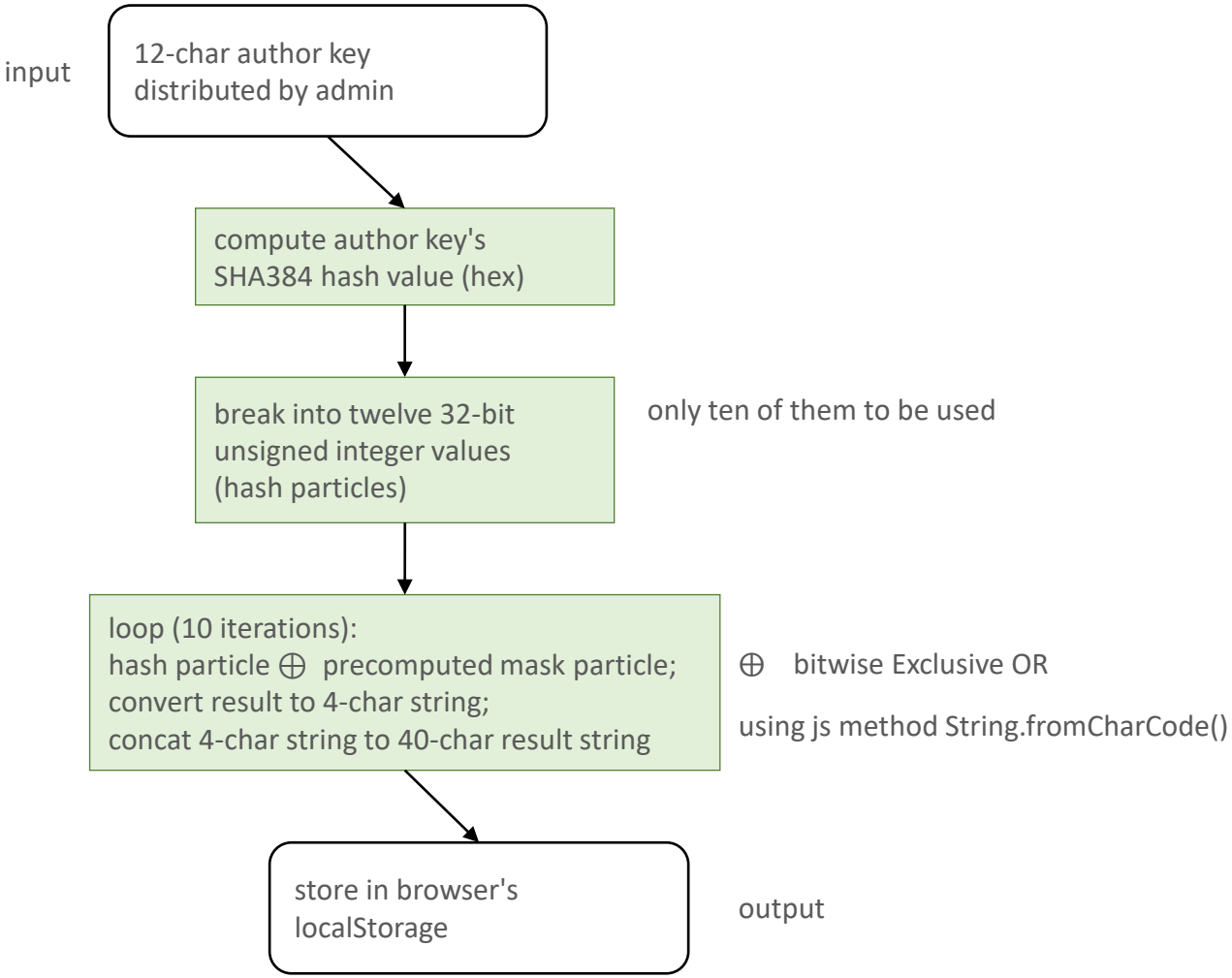
# Learn more about GitHub's Personal Access Token (PAT) format

- 40 characters string,
  starting with prefix: `ghp_`
- character set for remaining 36 chars
  result of base62 encoding, i.e.
  - capital letters A-Z
  - lower case letters a-z
  - numbers 0-9
- find more info here

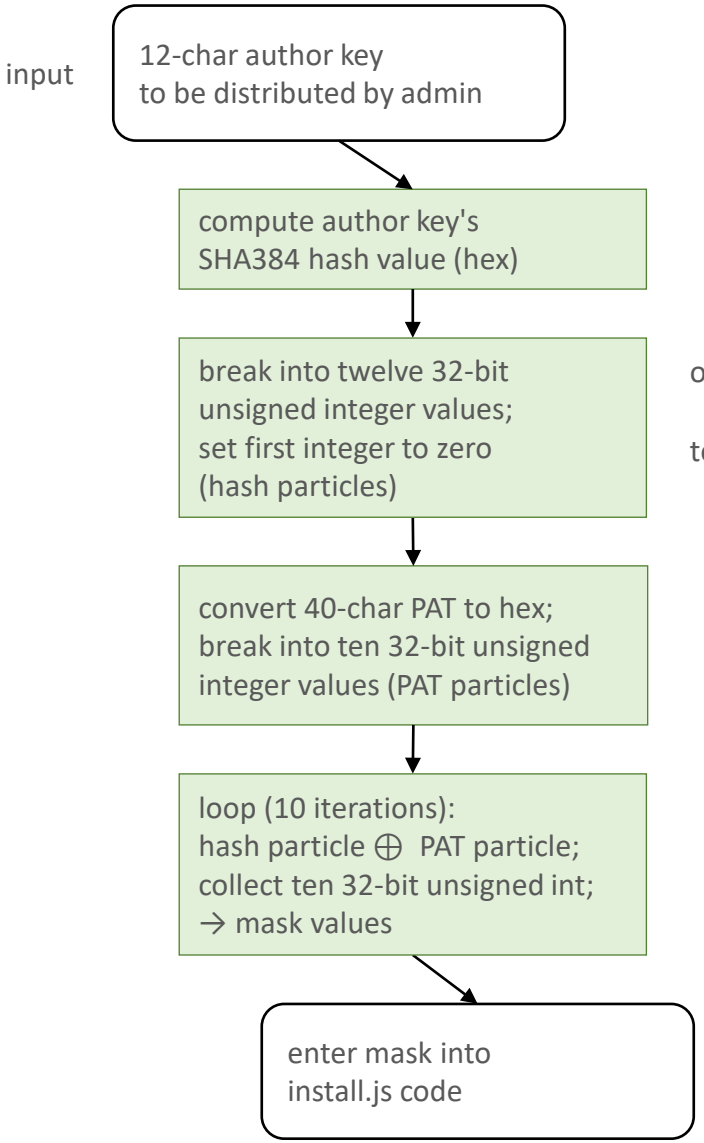


graphics from GitHub page

# PAT computation in install.js



# Mask computation




Remember:

EXOR is an involutory function;

thus: $A = B \oplus C \rightarrow B = C \oplus A$

only ten of them to be used

to preserve PAT prefix in EXOR operation below


frVocab

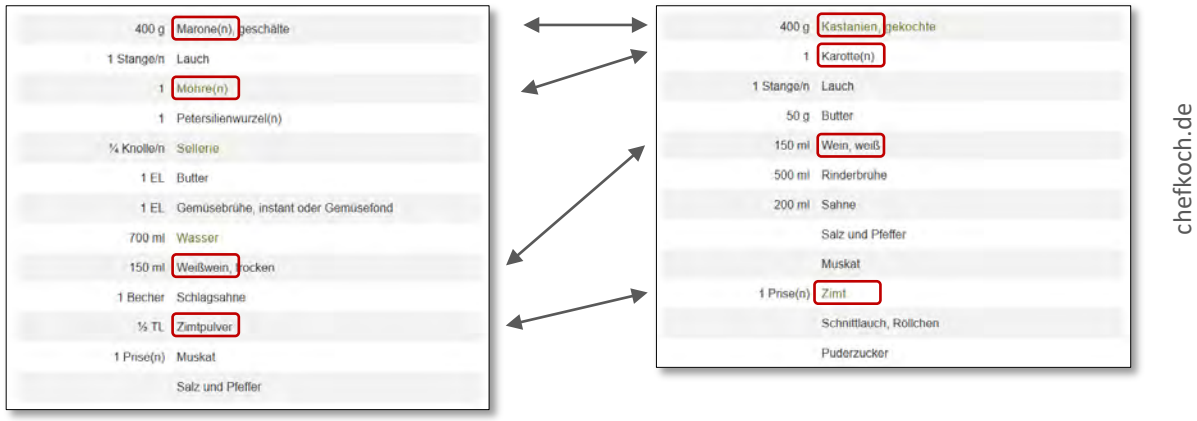
# What is a controlled vocabulary?

"A controlled vocabulary is an organized arrangement of words and phrases used to index content and/or to retrieve content through browsing or searching. It typically includes preferred and variant terms and has a defined scope or describes a specific domain."

https://www.getty.edu/research/publications/electronic_publications/intro_controlled_vocab/what.pdf

# frVocab purpose

Different recipes may use different names for same ingredients




chefkoch.de

- ingredient graph construction needs same names for same ingredients
- needed: controlled vocabulary of "normalized ingredient names"
- problem: derive normalized ingredient name from given ingredient name: "matching"
  - manually: high effort
  - automatically: errors nearly inevitable

  → semi-automatically

## *fruschtique controlled vocab for ingredient names*

- frVocab composed of two parts
  *ingredient classes* and *ingredients*
- *ingredients* part composed of one entry per ingredient
  - each entry holds a unique ID
  - each entry may hold several synonymes
  - each entry holds an ingredient class tag ("alc", "veg", "nut", etc.) for coloring of the ingredient graph nodes
  - additional information, e.g.
    label for ingredient graph nodes, and other
  - XML coding

# frVocab XML coding

Two parts: ingredient classes

```
<fc:classes>
  <fc:class classID="alc">
    <fc:classLabel>Alkohol</fc:classLabel>
    <fc:classColor>AliceBlue</fc:classColor>
    <fc:classColorHex>F0F8FF</fc:classColorHex>
    <fc:classColorRGBa>rgba( 38,131,131,1)</fc:classColorRGBa>
    <fc:R>240</fc:R>
    <fc:G>248</fc:G>
    <fc:B>255</fc:B>
    <fc:a>1</fc:a>
  </fc:class>
  <fc:class classID="carb">
    <fc:classLabel>Kohlenhydrate</fc:classLabel>
    <fc:classColor>BurlyWood</fc:classColor>
    <fc:classColorHex>DEB887</fc:classColorHex>
    <fc:classColorRGBa>rgba( 17,113,113,1)</fc:classColorRGBa>
    <fc:R>222</fc:R>
    <fc:G>184</fc:G>
    <fc:B>135</fc:B>
    <fc:a>1</fc:a>
  </fc:class>
  <fc:class classID="condi">
    <fc:classLabel>Zubereitung</fc:classLabel>
    <fc:classColor>CadetBlue</fc:classColor>
    <fc:classColorHex>5F9EA0</fc:classColorHex>
    <fc:classColorRGBa>rgba(  5, 93, 93,1)</fc:classColorRGBa>
    <fc:R>95</fc:R>
    <fc:G>158</fc:G>
    <fc:B>160</fc:B>
    <fc:a>1</fc:a>
  </fc:class>
  <fc:class classID="egg">
    <fc:classLabel>Ei</fc:classLabel>
    <fc:classColor>Orange</fc:classColor>
    <fc:classColorHex>FFA500</fc:classColorHex>
    <fc:classColorRGBa>rgba( 86,197, 86,1)</fc:classColorRGBa>
    <fc:R>255</fc:R>
    <fc:G>165</fc:G>
    <fc:B>0</fc:B>
    <fc:a>1</fc:a>
  </fc:class>
    ...
</fc:classes>
```

ingredient classes (16 in total):

meat, fish, herb, spice,
veg, fruit, onion, nuts,
carb, sweet, alc, condi,
mild, egg, fat, etc

# frVocab XML coding

Two parts: ingredients

approx. 430 entries

```
<fc:ingredients>
    ...
  <fc:ingredient id="lauch">
    <fc:igtLabel>Lauch</fc:igtLabel>
    <fc:igtSyn>porree</fc:igtSyn>
    <fc:igtClass>veg</fc:igtClass>
  </fc:ingredient>
  <fc:ingredient id="lauchzwiebel">
    <fc:igtLabel>Lauchzwiebel</fc:igtLabel>
    <fc:igtSyn>frühlingszwiebel</fc:igtSyn>
    <fc:igtClass>onion</fc:igtClass>
  </fc:ingredient>
  <fc:ingredient id="laugengebäck">
    <fc:igtLabel>Laugengebäck</fc:igtLabel>
    <fc:igtClass>carb</fc:igtClass>
  </fc:ingredient>
  <fc:ingredient id="leber">
    <fc:igtLabel>Leber</fc:igtLabel>
    <fc:igtClass>meat</fc:igtClass>
  </fc:ingredient>
  <fc:ingredient id="leberwurst">
    <fc:igtLabel>Leberwurst</fc:igtLabel>
    <fc:igtClass>meat</fc:igtClass>
  </fc:ingredient>
  <fc:ingredient id="lebkuchen">
    <fc:igtLabel>Lebkuchen</fc:igtLabel>
    <fc:igtClass>sweet</fc:igtClass>
  </fc:ingredient>
  <fc:ingredient id="liebstöckel">
    <fc:igtLabel>Liebstöckel</fc:igtLabel>
    <fc:igtClass>herb</fc:igtClass>
  </fc:ingredient>
    ...
</fc:ingredients>
```

frVocab entry

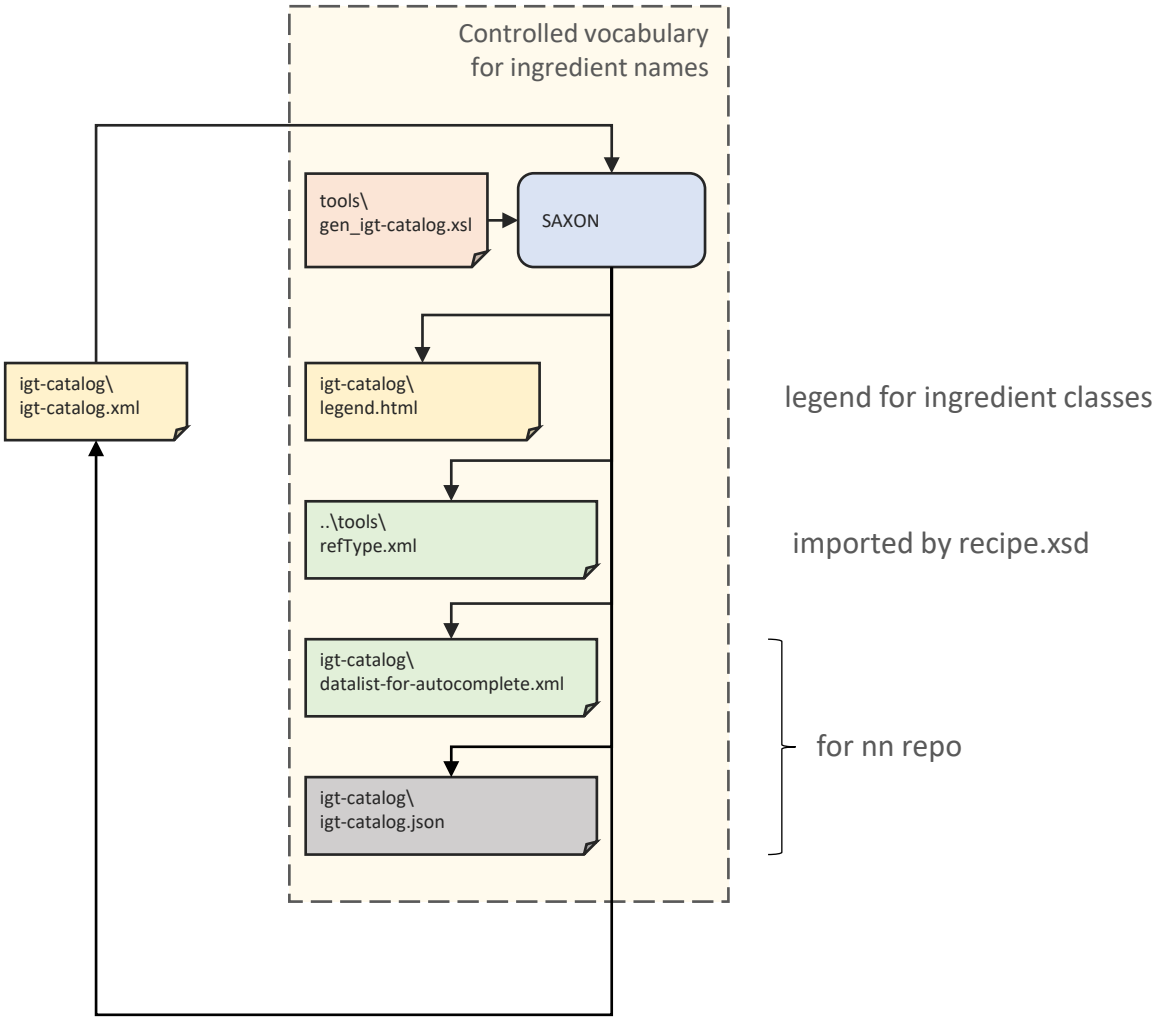
ID, serving as reference to frVocab entry

label for ingredient graph node

synonym (always lower case)

ingredient class, used for ingredient graph node coloring

# frVocab tool



# frVocab in nn repo

three sample entries:


```
{
  "id": "fleisch",
  "label": "Fleisch",
  "normName": "fleisch",
  "synonyms": []
},
{
  "id": "fleischbrühe",
  "label": "Fleischbrühe",
  "normName": "fleischbrühe",
  "synonyms": [
    "fleischfond",
    "rinderfond",
    "bouillon",
    "wildfond"
  ]
},
⋮
{
  "id": "saure_sahne",
  "label": "saure_Sahne",
  "normName": "saure sahne",
  "synonyms": []
},
```

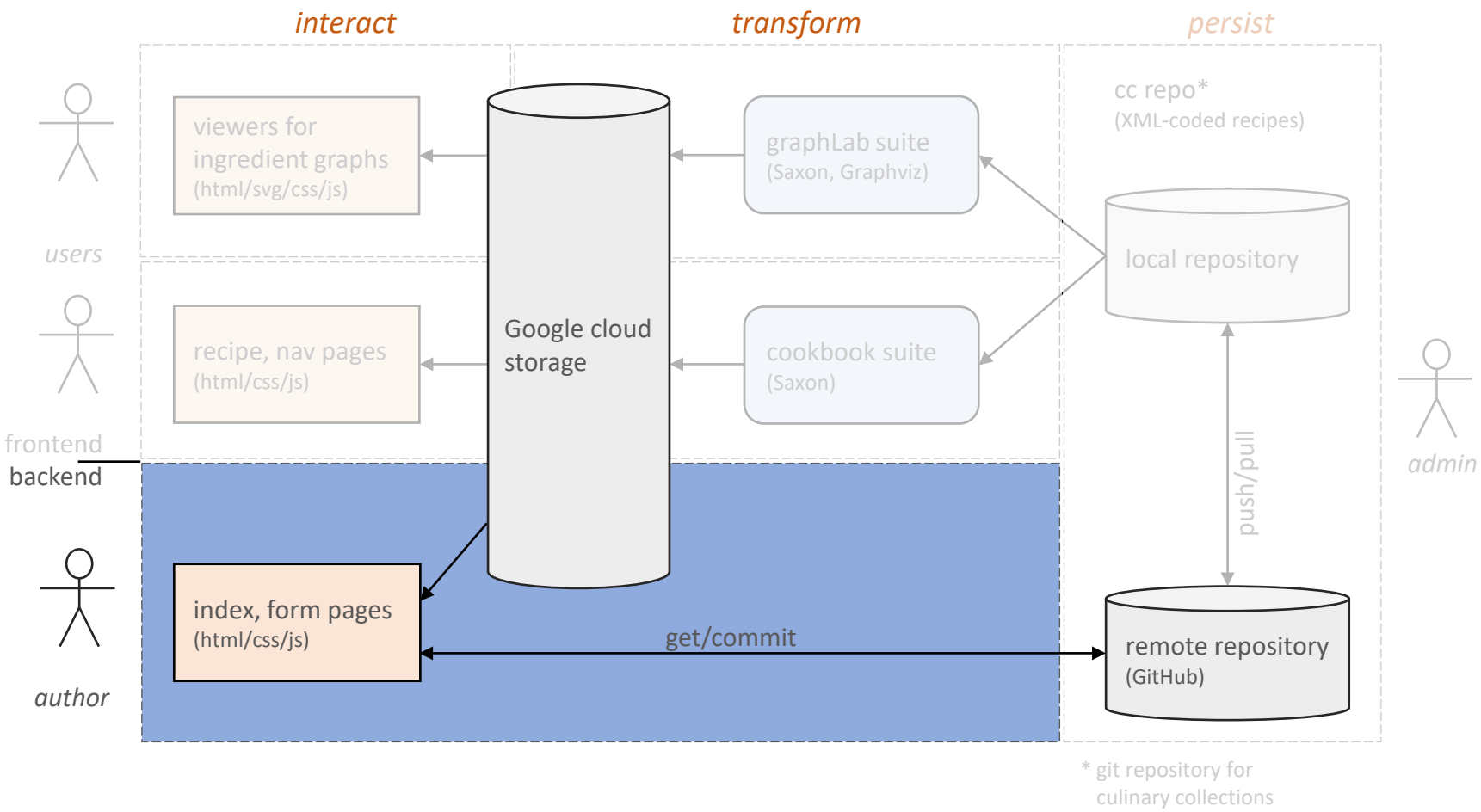

*normName* derived from ingredient ID:
replace "_" by " " (space)

## *fruschtique controlled vocab for ingredient names*

- matching `normName` with given ingredient name by string compare
  - convert given ingredient name to lower case
  - replace several special characters in given ingredient name appropriately, e.g. "(" to ""
- 3 indicators for match ranking
  - match position in word
  - match length
  - match with `normName` or with synonym


Forms

# fruschtique 3-tier architecture

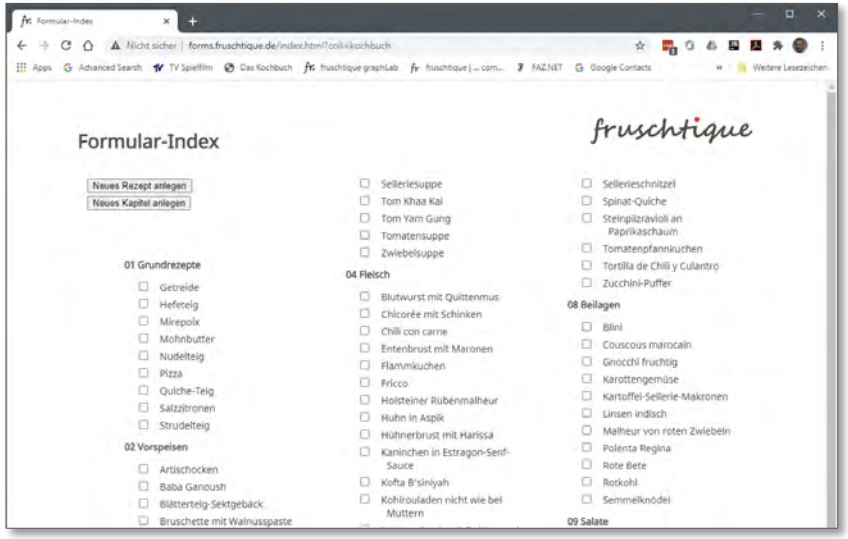


# Concept

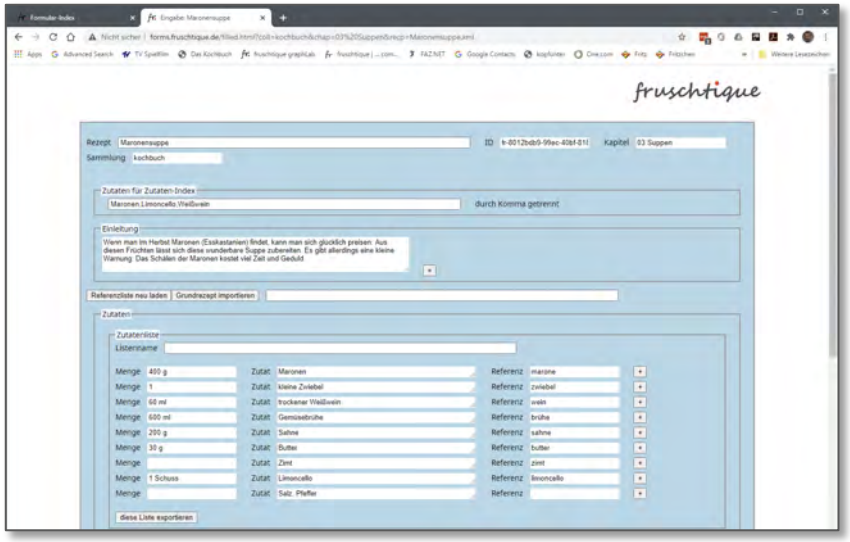
- Form skeletons supplied by Google cloud storage
- Full forms generated in-browser using cc repo content
- Get normalized ingredient names from
  *

* required for building ingredient graphs
(see graphLab section)

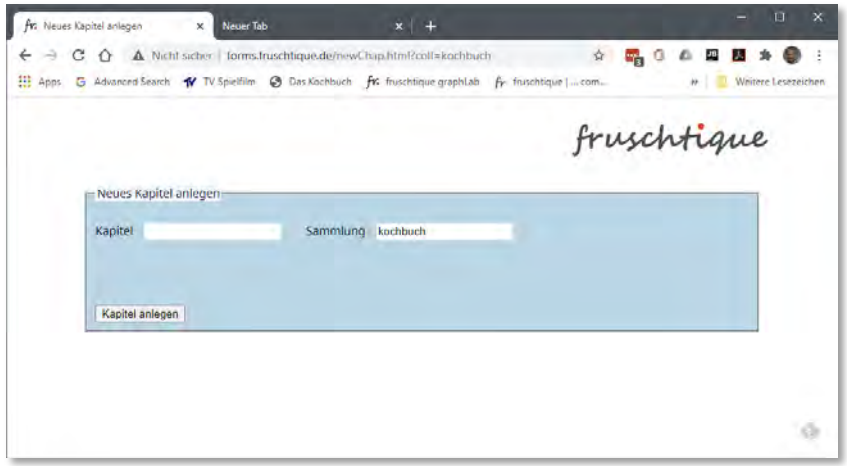
# 4 kinds of forms/pages



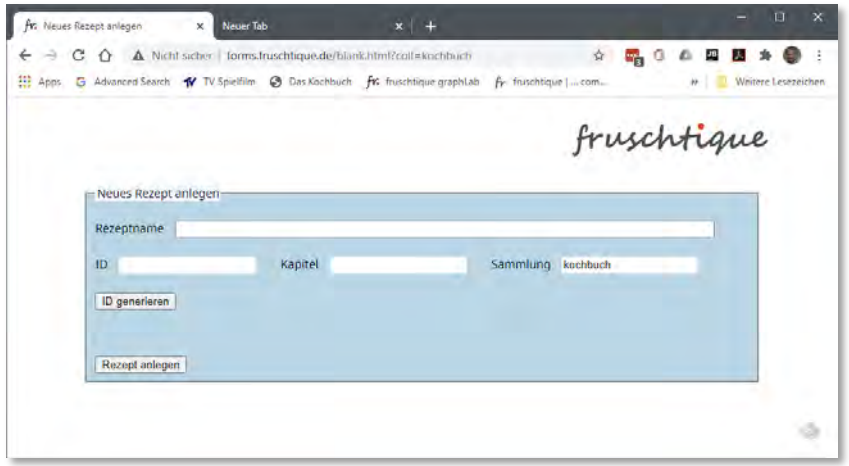

Forms index




Recipe form

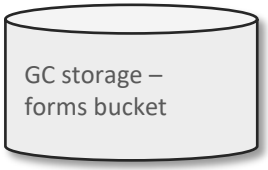



Register new chapter




Register new recipe

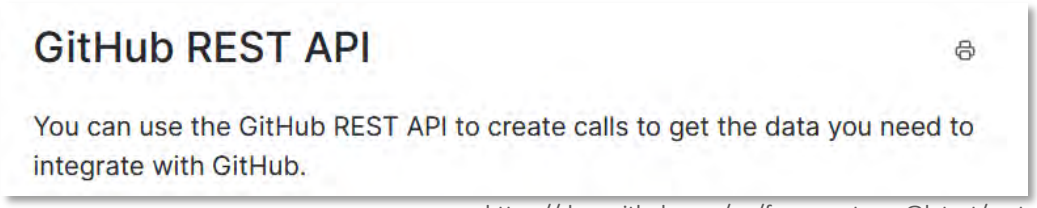
# Form skeletons




- index.html — form skeleton for collection index
- filled.html — form skeleton for existing recipes
- newChap.html — form skeleton for registering new chapters
- blank.html — form skeleton for registering new recipes
- …
- js
  - formsIndex.js
  - fillFormGit.js
  - fillBlankGit.js
  - …
- css
  - forms.css
  - …

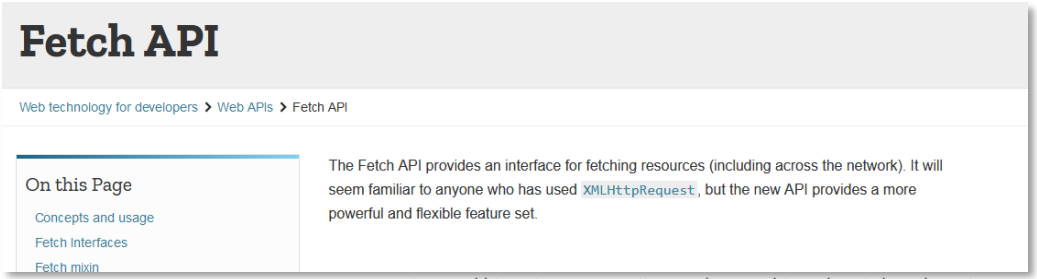
# Browser APIs involved in form generation

- GitHub REST API




https://docs.github.com/en/free-pro-team@latest/rest

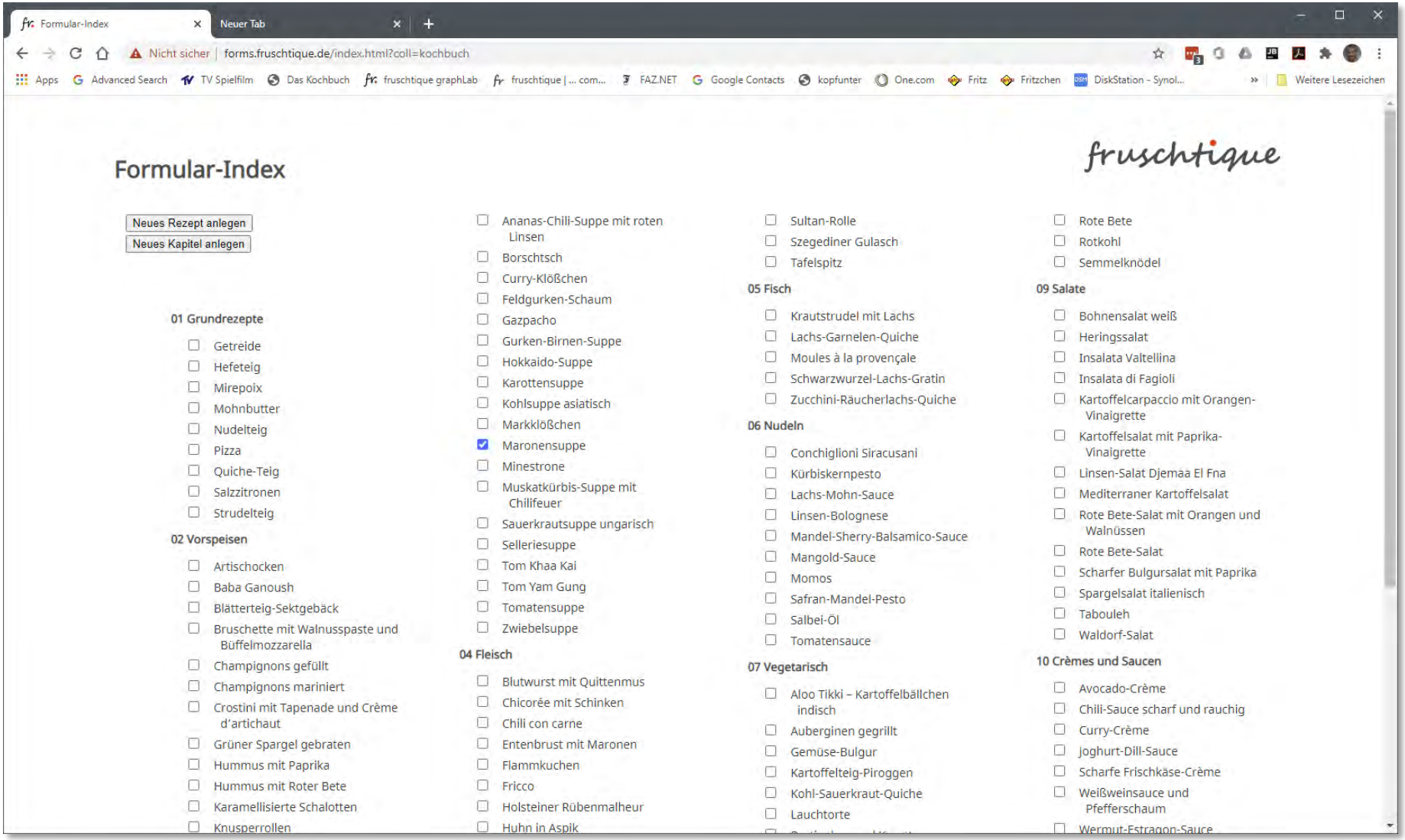
- Browser-native JS Fetch API




https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

In fruschtique, GitHub REST API calls are implemented by Fetch API calls.

# Forms index


Formular-Index
Neuer Tab
Nicht sicher | forms.fruschtique.de/index.html?coll=kochbuch
Apps Advanced Search TV Spielfilm Das Kochbuch fruschtique graphLab fruschtique | ... com... FAZ.NET Google Contacts kopfunter One.com Fritz Fritzchen DiskStation - Synol... Weitere Lesezeichen
Formular-Index
fruschtique
Neues Rezept anlegen
Neues Kapitel anlegen
01 Grundrezepte
Getreide
Hefeteig
Mirepoix
Mohnbutter
Nudelteig
Pizza
Quiche-Teig
Salzzitronen
Strudelteig
02 Vorspeisen
Artischocken
Baba Ganoush
Blätterteig-Sektgebäck
Bruschette mit Walnusspaste und Büffelmozzarella
Champignons gefüllt
Champignons mariniert
Crostini mit Tapenade und Crème d'artichaut
Grüner Spargel gebraten
Hummus mit Paprika
Hummus mit Roter Bete
Karamellisierte Schalotten
Knusperrollen
Ananas-Chili-Suppe mit roten Linsen
Borschtsch
Curry-Klößchen
Feldgurken-Schaum
Gazpacho
Gurken-Birnen-Suppe
Hokkaido-Suppe
Karottensuppe
Kohlsuppe asiatisch
Markklößchen
Maronensuppe
Minestrone
Muskatkürbis-Suppe mit Chilifeuer
Sauerkrautsuppe ungarisch
Selleriesuppe
Tom Khaa Kai
Tom Yam Gung
Tomatensuppe
Zwiebelsuppe
04 Fleisch
Blutwurst mit Quittenmus
Chicorée mit Schinken
Chili con carne
Entenbrust mit Maronen
Flammkuchen
Fricco
Holsteiner Rübenmalheur
Huhn in Aspik
Sultan-Rolle
Szegediner Gulasch
Tafelspitz
05 Fisch
Krautstrudel mit Lachs
Lachs-Garnelen-Quiche
Moules à la provençale
Schwarzwurzel-Lachs-Gratin
Zucchini-Räucherlachs-Quiche
06 Nudeln
Conchiglioni Siracusani
Kürbiskernpesto
Lachs-Mohn-Sauce
Linsen-Bolognese
Mandel-Sherry-Balsamico-Sauce
Mangold-Sauce
Momos
Safran-Mandel-Pesto
Salbei-Öl
Tomatensauce
07 Vegetarisch
Aloo Tikki – Kartoffelbällchen indisch
Auberginen gegrillt
Gemüse-Bulgur
Kartoffelteig-Piroggen
Kohl-Sauerkraut-Quiche
Lauchtorte
Rote Bete
Rotkohl
Semmelknödel
09 Salate
Bohnensalat weiß
Heringssalat
Insalata Valtellina
Insalata di Fagioli
Kartoffelcarpaccio mit Orangen-Vinaigrette
Kartoffelsalat mit Paprika-Vinaigrette
Linsen-Salat Djemaa El Fna
Mediterraner Kartoffelsalat
Rote Bete-Salat mit Orangen und Walnüssen
Rote Bete-Salat
Scharfer Bulgursalat mit Paprika
Spargelsalat italienisch
Tabouleh
Waldorf-Salat
10 Crèmes und Saucen
Avocado-Crème
Chili-Sauce scharf und rauchig
Curry-Crème
Joghurt-Dill-Sauce
Scharfe Frischkäse-Crème
Weißweinsauce und Pfefferschaum
Wermut-Estragon-Sauce

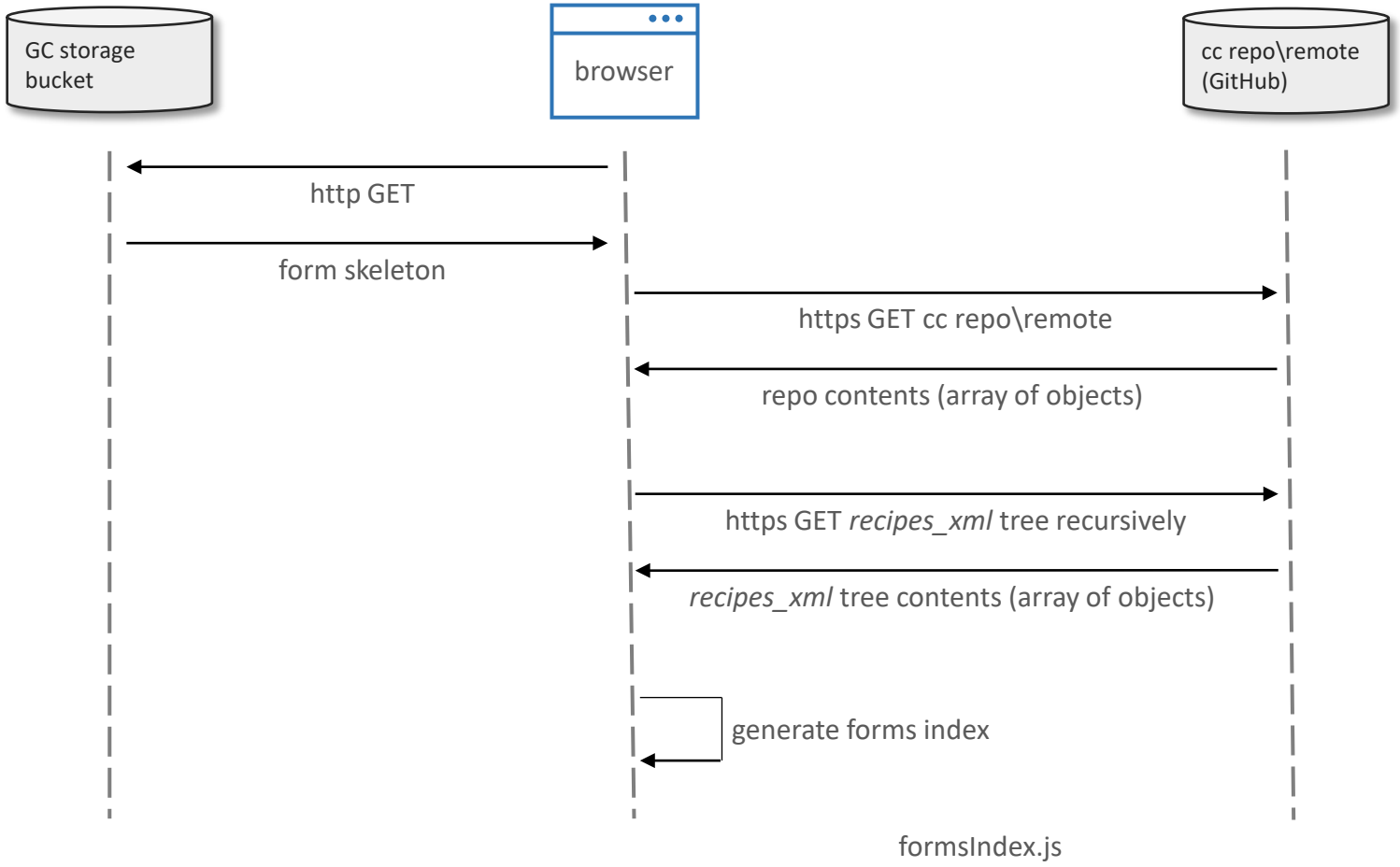
# Forms index: sample URL

http://forms.fruschtique.de/index.html?coll=kochbuch

form skeleton

parameter for collection selection

# Forms index generation



# Retrieve remote repository index

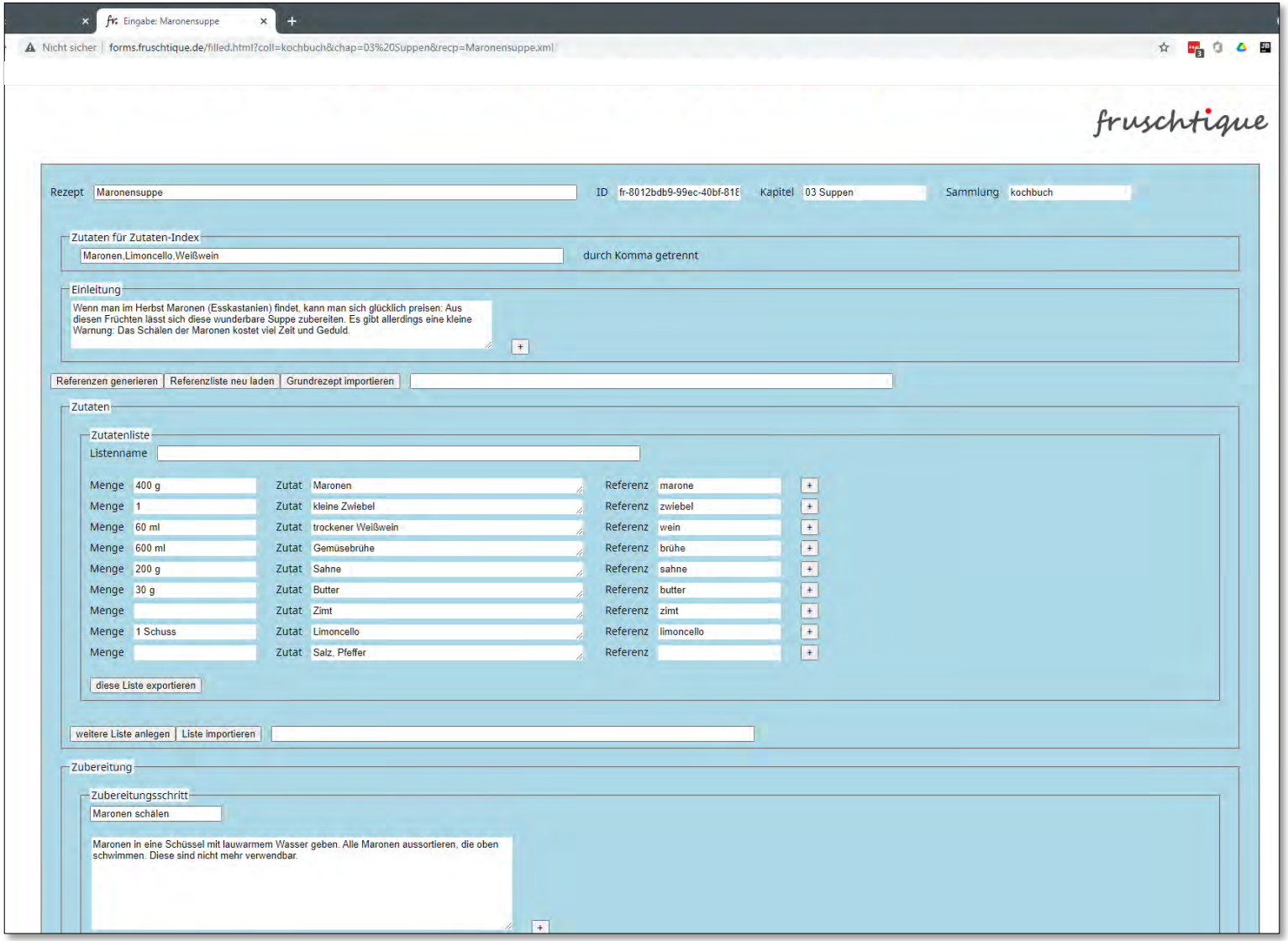
```
hdrs = {
  'Accept': 'application/vnd.github.v3+json',
  'Authorization': apiKey
}
// create list of recipes
let url_str;
url_str = `https://api.github.com/repos/nluttenberger/${myColl}/contents`;
fetch(url_str,{headers: hdrs})
  .then(resp => {
    return resp.json();
    })
  .then(data => {
    let ix = data.indexOf(data.filter(function(item) {
      return item.path === "recipes_xml"
    })[0])
    let sha = data[ix].sha;
    url_str = `https://api.github.com/repos/nluttenberger/${myColl}/git/trees/${sha}?recursive=true`;
    fetch(url_str,{headers: hdrs})
      .then (resp =>  {
        console.log('Sammlungsindex eingelesen: ', resp.status, resp.statusText);
        return resp.json()
      })
      .then(data => {
        let tree = data.tree;
        listHTML (tree);
      })
      .catch ((error) => {
        console.log('Error while reading directory listings:', error);
      })
    })
  .catch ((error) => {
    console.log('Error while reading collection sha:', error);
  })
```

get *recipes_xml* as Git tree sha

get tree

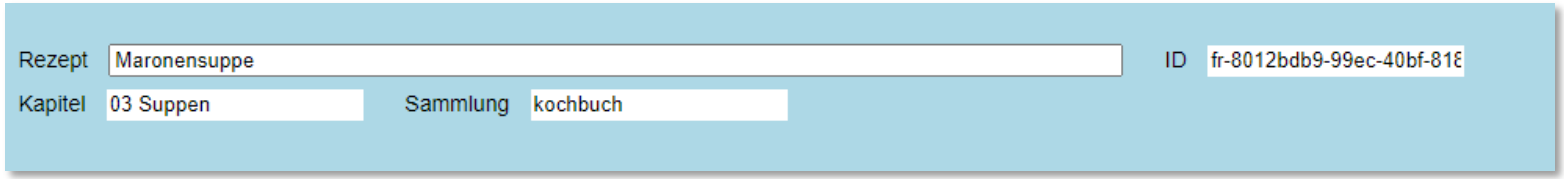
formsIndex.js

# Recipe form


Eingabe: Maronensuppe
Nicht sicher
forms.fruschtique.de/filled.html?coll=kochbuch&chap=03%20Suppen&recp=Maronensuppe.xml
fruschtique
Rezept
Maronensuppe
ID
fr-8012bdb9-99ec-40bf-81
Kapitel
03 Suppen
Sammlung
kochbuch
Zutaten für Zutaten-Index
Maronen,Limoncello,Weißwein
durch Komma getrennt
Einleitung
Wenn man im Herbst Maronen (Esskastanien) findet, kann man sich glücklich preisen: Aus diesen Früchten lässt sich diese wunderbare Suppe zubereiten. Es gibt allerdings eine kleine Warnung: Das Schälen der Maronen kostet viel Zeit und Geduld.
Referenzen generieren
Referenzliste neu laden
Grundrezept importieren
Zutaten
Zutatenliste
Listenname
Menge 400 g Zutat Maronen Referenz marone
Menge 1 Zutat kleine Zwiebel Referenz zwiebel
Menge 60 ml Zutat trockener Weißwein Referenz wein
Menge 600 ml Zutat Gemüsebrühe Referenz brühe
Menge 200 g Zutat Sahne Referenz sahne
Menge 30 g Zutat Butter Referenz butter
Menge Zutat Zimt Referenz zimt
Menge 1 Schuss Zutat Limoncello Referenz limoncello
Menge Zutat Salz, Pfeffer Referenz
diese Liste exportieren
weitere Liste anlegen
Liste importieren
Zubereitung
Zubereitungsschritt
Maronen schälen
Maronen in eine Schüssel mit lauwarmem Wasser geben. Alle Maronen aussortieren, die oben schwimmen. Diese sind nicht mehr verwendbar.

# Recipe form: sample URL

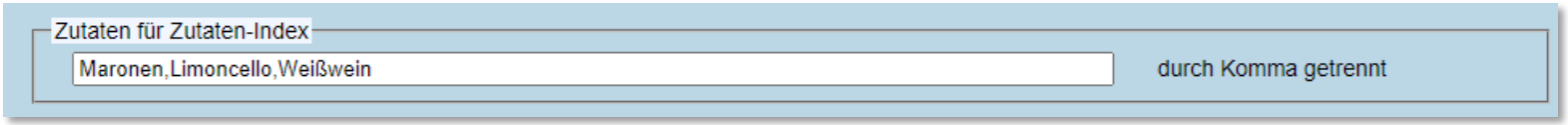
# Recipe form highlights




Each recipe has

- name (reflected in XML file name, in image name, and in thumb name)
- chapter
- collection
- ID (generated or user-defined, usage → see later)

# Recipe form highlights (2)



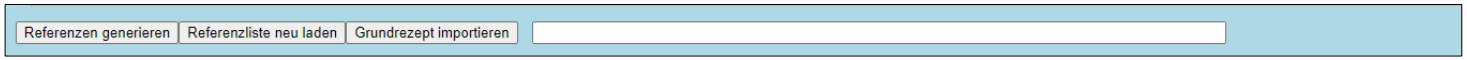

Key ingredients

(see *cookbook suite* section)

# Recipe form highlights (3)

*Recipe import*

Some recipes have references to other recipes, e.g. a pizza recipe refering to a pizza dough recipe




name of recipe to be imported (auto-complete)

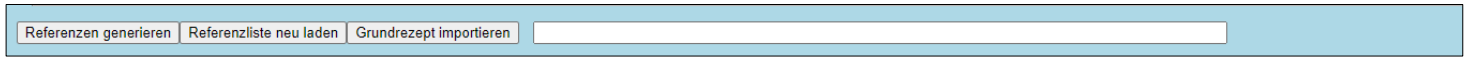
Import recipe from selected import chapter (determined in `descriptor.json`)

`descriptor.json` example:

```
{"importFrom": "recipes_xml/00 Allgemeine Regeln"}
```

# Recipe form highlights (4)

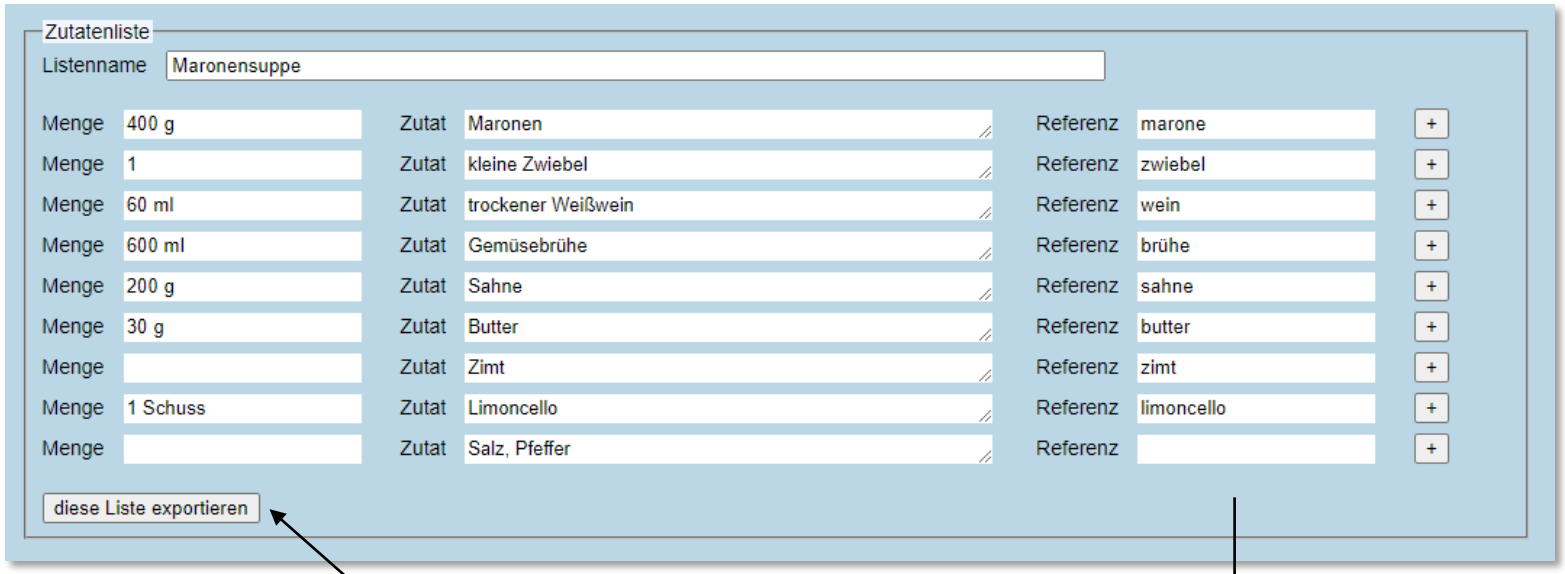
Interface to *fruschtique controlled vocabulary for ingredient names*




Re-load HTML auto-complete list of the
*fruschtique controlled vocabulary for ingredient names*,
e.g. after vocabulary update

Try to derive a matching entry in the
*fruschtique controlled vocabulary for ingredient names*
from the given ingredient names (for all ingredients in recipe)

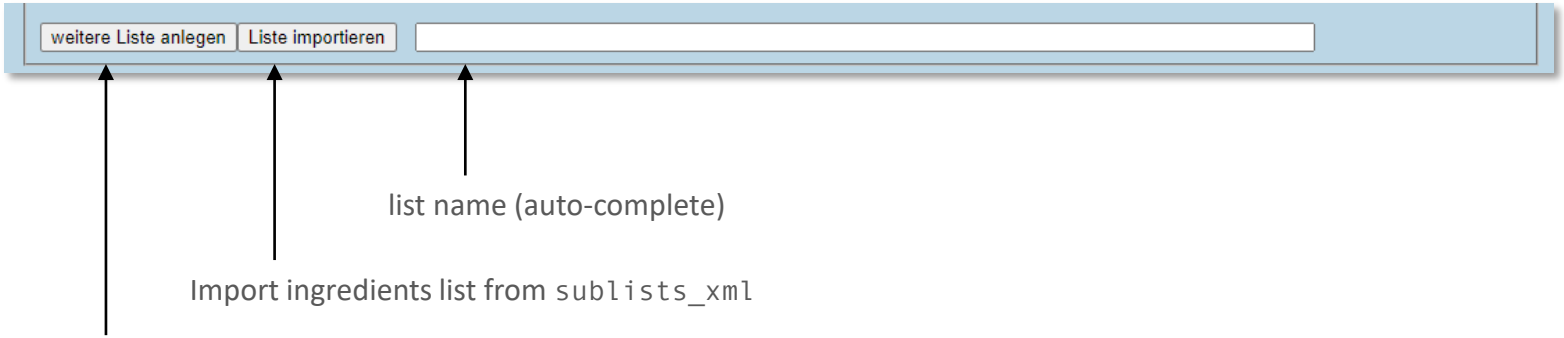
# Recipe form highlights (5)




Named ingredients lists may be exported to
*sublists_xml* tree in cc repo
for later import into other recipe

references to the
*fruschtique controlled vocabulary*
*for ingredient names*

# Recipe form highlights (6)



# Retrieve recipe xml file

```
hdrs = {
  'Accept': 'application/vnd.github.v3+json',
  'Authorization': apiKey
}
let myURL = new URL (document.URL);
myColl = myURL.searchParams.get('coll');
myChap = myURL.searchParams.get('chap');
myRecp = myURL.searchParams.get('recp');
console.log (`Sammlung: ${myColl}, Kapitel: ${myChap}, Rezept: ${myRecp}`);
let url_str = `https://api.github.com/repos/nluttenberger/${myColl}/contents/recipes_xml/${myChap}/${myRecp}`;
fetch (url_str,{headers: hdrs})
  .then (resp => resp.json())
  .then (data => {
    rcpXML = b64_to_utf8(data.content);
    gitName = data.name;
    gitPath = data.path;
    gitSHA = data.sha;
    makeForm(rcpXML);
  })
  .catch ((error) => {
    console.log('Error while reading recipe xml data:', error);
  })
```

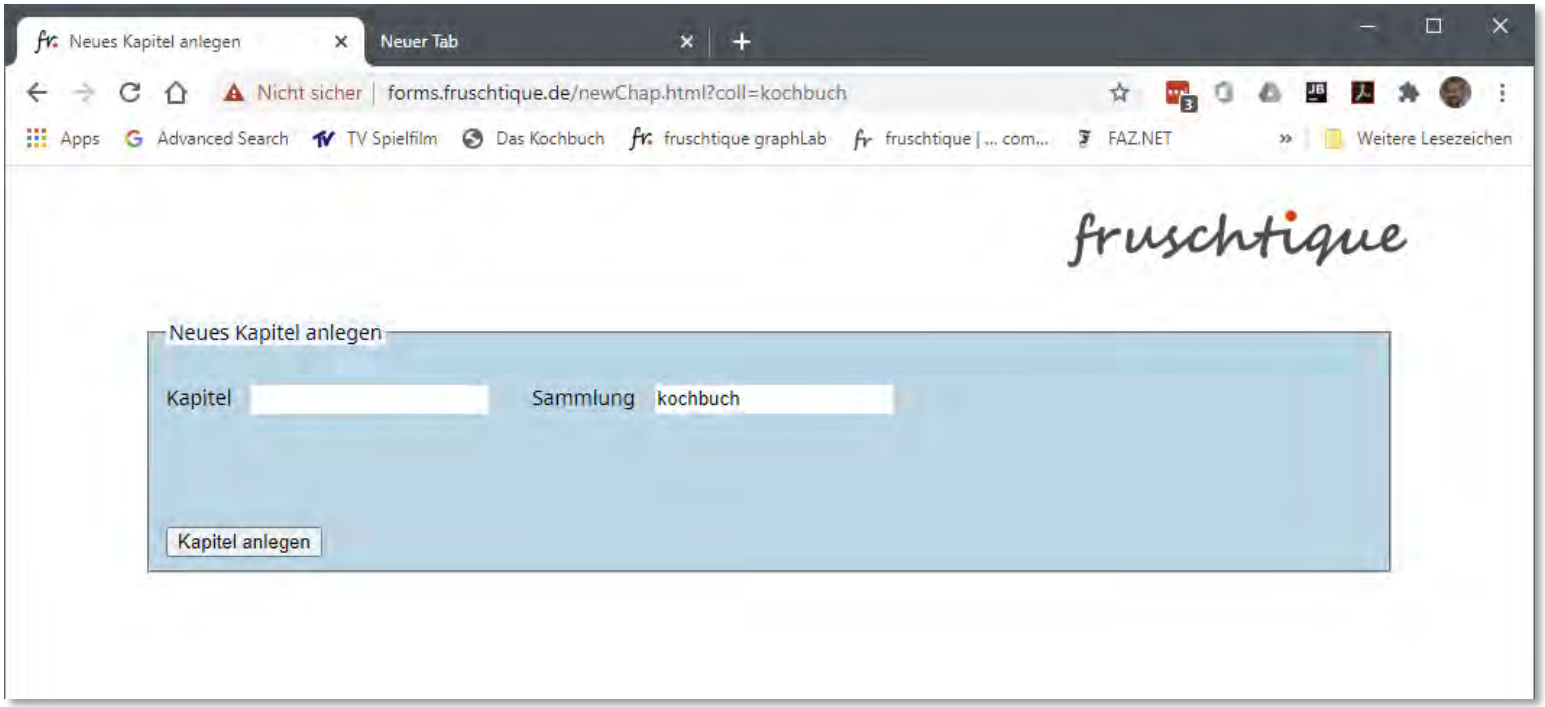
fillFormGit.js

# Commit xml file to cc repo\remote

```
let xmlText = new XMLSerializer().serializeToString(xmlDoc);
// convert updated recipe to base64 ---------------------------------
let b64Recipe = utf8_to_b64(xmlText);
//build update object and url ---------------------------------------
let update = {
  'message': 'update',
  'content': b64Recipe,
  'sha': gitSHA
}
let urlStr = `https://api.github.com/repos/nluttenberger/${myColl}/contents/${gitPath}`;
// upload and commit ------------------------------------------------
fetch (urlStr,{
  method: 'PUT',
  body: JSON.stringify(update),
  headers: hdrs
  })
  .then (resp => {
    console.log('Update: ', resp.status, resp.statusText);
    if (resp.status === 200) {
      alert ('Rezept abgespeichert!')
      location.reload(true);
    }
    return resp.json()
  })
  .then (data => {
    //console.log (data.commit);
  })
  .catch((error) => {
    console.error('Error while saving recipe: ', error);
})
```

fillFormGit.js

# Register new chapter




Sample url:
http://forms.fruschtique.de/newChap.html?coll=kochbuch

Chapter names start with two decimal digits, followed by a blank!

# Register new chapter to cc repo\remote


An easy way to create a new directory (i.e. a new chapter) in a Git repository is to store at least one file in the directory to be created. To meet this procedure, the fruschtique JS function for registering a new chapter creates a file *[chaptername].xml* and stores it under *recipes_xml/[chaptername]/[chaptername].xml*.
This "hack" happily matches the fruschtique Cookbook suite rules (see below).

The fruschtique Cookbook suite assumes a file *[chaptername].xml* in each chapter. It is used by the Cookbook suite to generate a chapterwise table of contents. The file format follows the recipe XML schema. The *<fr:recipeName>*, *<fr:book>*, and *<fr:chapter>* elements must have appropriate text values.*

fruschtique

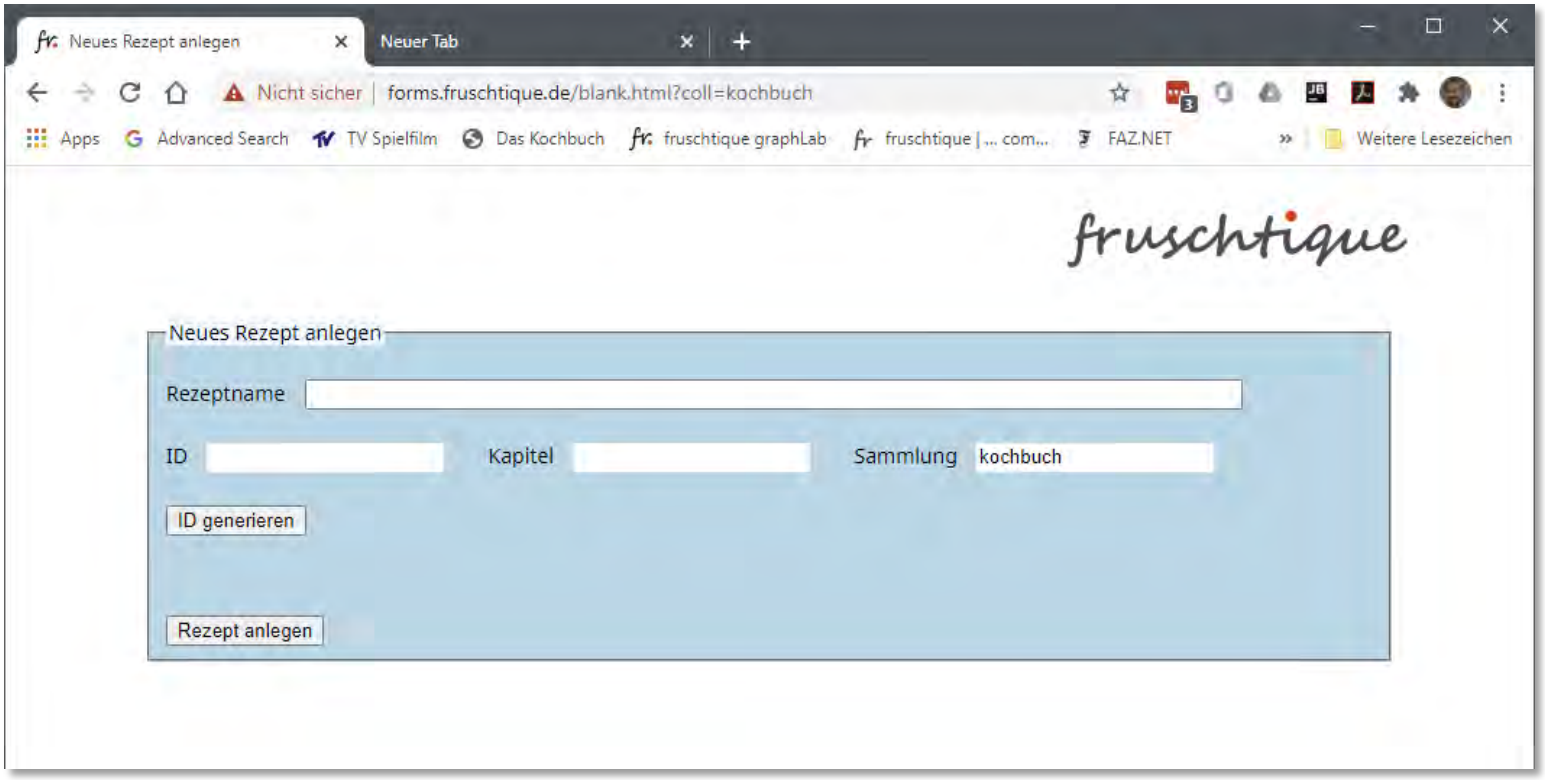
* see *Cookbook suite* section

# Register new chapter

```
let parser = new DOMParser();
xmlDoc = parser.parseFromString(text, "text/xml");
let xmlText = new XMLSerializer().serializeToString(xmlDoc);
// convert updated recipe to base64 ----------------------------------
let b64Recipe = utf8_to_b64(xmlText);
//build update object and url ---------------------------------------
let update = {
  'message': 'just created',
  'content': b64Recipe
}
let urlStr = `https://api.github.com/repos/nluttenberger/${myColl}/contents/recipes_xml/${myChap}/${myRecp}.xml`;
// upload and commit -------------------------------------------------
fetch (urlStr,{
  method: 'PUT',
  body: JSON.stringify(update),
  headers: hdrs
})
  .then (resp => {
    if (resp.status === 201) {
      alert ('Kapitel angelegt!')
    }
    return resp.json()
  })
  .then (data => {
    console.log (data.commit);
  })
  .catch((error) => {
    console.error('Error while creating chapter: ', error);
  })
```

fillBlankGit.js

# Register new recipe




Sample url:
http://forms.fruschtique.de/blank.html?coll=kochbuch

# Register new recipe

```
//--- create empty form ---------------------------------------------
createRecipe();
//--- serialize form input to XML -----------------------------------------
let xmlText = new XMLSerializer().serializeToString(xmlDoc);
// convert updated recipe to base64 ----------------------------------
let b64Recipe = utf8_to_b64(xmlText);
//build update object and url ----------------------------------------
let update = {
  'message': 'just created',
  'content': b64Recipe
}
let urlStr = `https://api.github.com/repos/nluttenberger/${myColl}/contents/recipes_xml/${myChap}/${myRecp}.xml`;
// upload and commit ---------------------------------------------------
fetch (urlStr,{
  method: 'PUT',
  body: JSON.stringify(update),
  headers: hdrs
  })
  .then (resp => {
    if (resp.status === 201) {
      alert ('Rezept angelegt!')
      let htmlStr = `<a id="toFullForm" href="filled.html?coll=${myColl}&chap=${myChap}&recp=${myRecp}.xml"></a>`
      $('body').append(htmlStr);
      document.getElementById('toFullForm').click();
    }
    return resp.json()
  })
  .then (data => {
    console.log (data.commit);
  })
  .catch((error) => {
    console.error('Error while saving recipe: ', error);
  })
```

click to full form

fillBlankGit.js

# Handling a new recipe

# Deleting a recipe

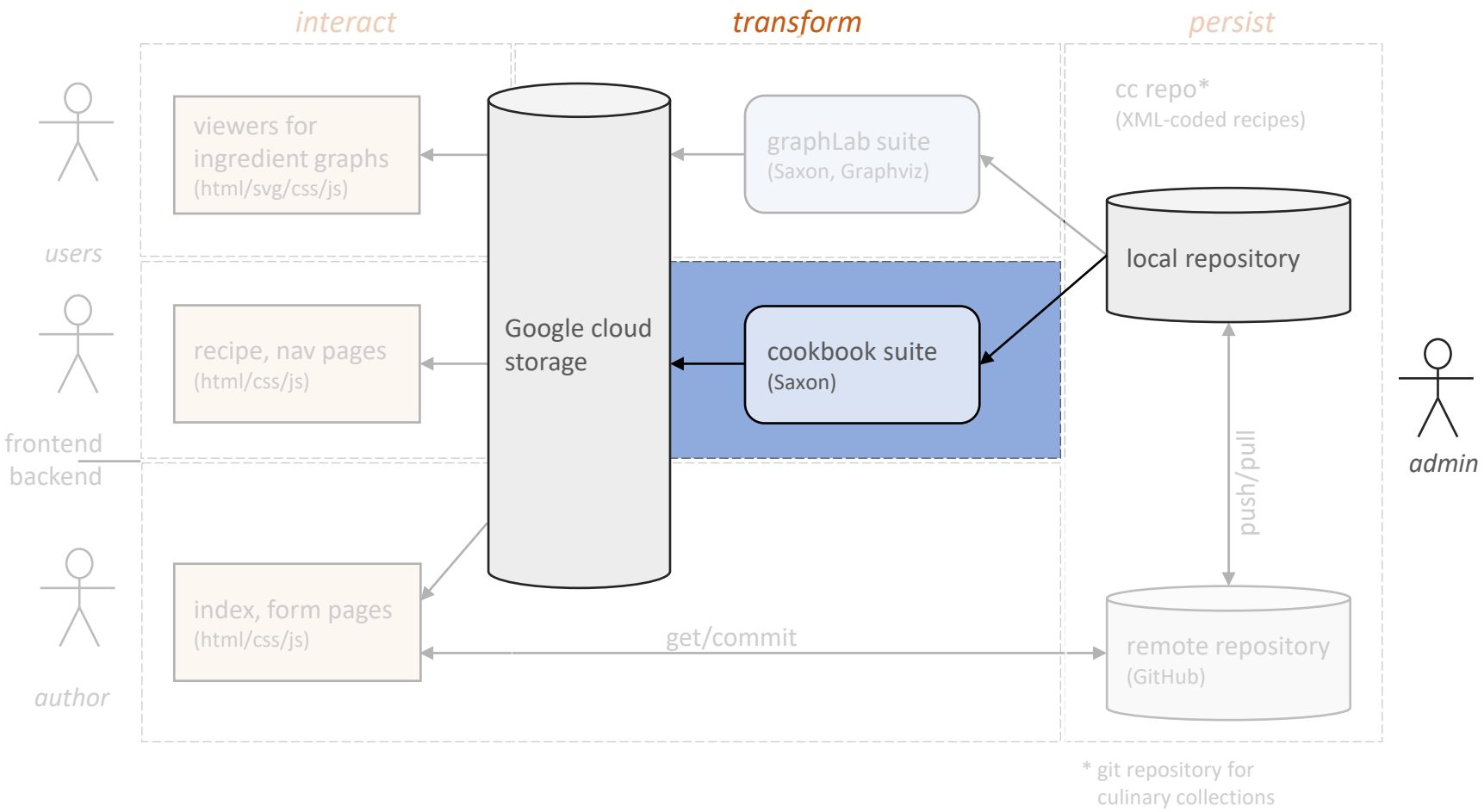
In GitHub Desktop:
→ `Show in Explorer`

Delete recipe file from `recipes_xml\[chapter]`

In GitHub Desktop:
→ `Commit to main` and → `Push origin`


Cookbook suite

# fruschtique 3-tier architecture



# Concept

- Generate a variety of navigation facilities offline
- Provide additional material about the collection
- Support subset of markdown syntax contained in XML text content

in more detail here

# XML schema for recipes

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| <?xml | version="1.0" encoding="UTF-8" | | | | | | | | |
| xs:schema | @targetNamespace | http://fruschtique.de/ns/recipe | | | | | | | |
| | @xmlns | http://fruschtique.de/ns/recipe | | | | | | | |
| | @xmlns:xs | http://www.w3.org/2001/XMLSchema | | | | | | | |
| | @xmlns:xsi | http://www.w3.org/2001/XMLSchema-instance | | | | | | | |
| | @elementFormDefault | qualified | | | | | | | |
| | xs:include | @schemaLocation | refType.xsd | | | | | | |
| | xs:complexType (7 rows) | | | | | | | | |
| | xs:element (2 rows) | @name | xs:complexType | | | | | | |
| | | 1 recipe | xs:complexType | xs:sequence | xs:element (10 rows) | @name | @type | @minOccurs | |
| | | | | | | 1 meta | metaType | 0 | |
| | | | | | | 2 recipeName | xs:string | | |
| | | | | | | 3 recipeKeywords | xs:token | 0 | |
| | | | | | | 4 recipeIntro | recipeIntroType | | |
| | | | | | | 5 recipeIngredients | recipeIngredientsType | | |
| | | | | | | 6 recipeInstructions | recipeInstructionsType | | |
| | | | | | | 7 recipeSideDish | xs:string | 0 | |
| | | | | | | 8 recipeOrigin | xs:string | 0 | |
| | | | | | | 9 recipeSeeAlso | xs:string | 0 | |
| | | | | | | 10 recipeLicense | xs:string | 0 | |
| | | | | | xs:attribute | @name | rcpID | | |
| | | | | | | @type | xs:ID | | |
| | | 2 descriptor | xs:complexType | | | | | | |

# Additional DTD for HTML entities

HTML entities acceptable in XML element text content

```
<!ENTITY nbsp " " >
<!ENTITY ndash "&#8211;" >
<!ENTITY uarr "&#8593;" >
<!ENTITY quot "&#34;" >
<!ENTITY hellip "&#x2026;" >
<!ENTITY frac12 "&#x00BD;" >
<!ENTITY frac14 "&#x00BC;" >
<!ENTITY frac34 "&#x00BE;" >
<!ENTITY deg "&#x00B0;" >
<!ENTITY sup2 "&#x00B2;" >
```

declared in entities.dtd

# Sample recipe

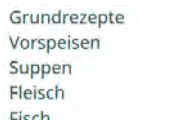
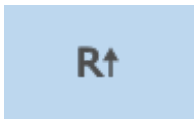
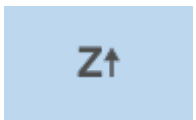
```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE stylesheet SYSTEM "file:///c:/users/nlutt/documents/Websites/tools/entities.dtd">
<fr:recipe xmlns:fr="http://fruschtique.de/ns/recipe" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" rcpID="fr-22ca9a38-ea72-41b9-82eb-f968034bb8c4"
xsi:schemaLocation="http://fruschtique.de/ns/recipe ../../../tools/recipe.xsd">
  <fr:meta>
    <fr:book>kochbuch</fr:book>
    <fr:chapter>02 Vorspeisen</fr:chapter>
  </fr:meta>
  <fr:recipeName>Lachs-Tatar mit grünem Spargel</fr:recipeName>
  <fr:recipeKeywords>Lachs,Grüner Spargel,Miso</fr:recipeKeywords>
  <fr:recipeIntro>
    <fr:introText>Eine einfach zuzubereitende Vorspeise, die hervorragend schmeckt.</fr:introText>
  </fr:recipeIntro>
  <fr:recipeIngredients>
    <fr:igdtList>
      <fr:igdtListName/>
      <fr:igdtListLine>
        <fr:igdtQuantity>500 g</fr:igdtQuantity>
        <fr:igdtName ref="spargel">grüner Spargel</fr:igdtName>
      </fr:igdtListLine>
      <fr:igdtListLine>
        <fr:igdtQuantity/>
        <fr:igdtName ref="sesam">Sesamöl</fr:igdtName>
      </fr:igdtListLine>
      <fr:igdtListLine>
        <fr:igdtQuantity>300 g</fr:igdtQuantity>
        <fr:igdtName ref="lachs">Lachsfilet ohne Haut (Sushi-Qualität)</fr:igdtName>
      </fr:igdtListLine>
      <fr:igdtListLine>
        <fr:igdtQuantity>5 EL</fr:igdtQuantity>
        <fr:igdtName ref="mirin">Mirin</fr:igdtName>
      </fr:igdtListLine>
      <fr:igdtListLine>
        <fr:igdtQuantity>1 EL</fr:igdtQuantity>
        <fr:igdtName ref="miso">Miso-Paste</fr:igdtName>
      </fr:igdtListLine>
      <fr:igdtListLine>
        <fr:igdtQuantity>½ EL</fr:igdtQuantity>
        <fr:igdtName ref="zucker">Puderzucker</fr:igdtName>
      </fr:igdtListLine>
      <fr:igdtListLine>
        <fr:igdtQuantity/>
        <fr:igdtName ref="chili">Chilipulver</fr:igdtName>
      </fr:igdtListLine>
      <fr:igdtListLine>
        <fr:igdtQuantity/>
        <fr:igdtName ref="">Salz</fr:igdtName>
      </fr:igdtListLine>
```

# Sample recipe (2)

```
<fr:igdtListLine>
          <fr:igdtQuantity/>
          <fr:igdtName ref="koriandergrün">Koriandergrün</fr:igdtName>
        </fr:igdtListLine>
      </fr:igdtList>
    </fr:recipeIngredients>
    <fr:recipeInstructions>
      <fr:instruction>
        <fr:instrStepName>Lachs-Tatar</fr:instrStepName>
        <fr:instrStepText>Gräten aus dem Lachsfilet entfernen. Das Filet in sehr kleine Würfel schneiden.</fr:instrStepText>
        <fr:instrStepText>Miso-Paste, Mirin, Puderzucker und Chili miteinander verrühren. Ggf. mit etwas Salz abschmecken. Etwas Koriandergrün hacken und ebenfalls in die Marinade einrühren.
Fischwürfel in die Marinade geben und ca. 60 min im Kühlschrank marinieren lassen. </fr:instrStepText>
      </fr:instruction>
      <fr:instruction>
        <fr:instrStepName>Spargel</fr:instrStepName>
        <fr:instrStepText>Den grünen Spargel waschen. Den unteren Teil großzügig abschneiden. Den Spargel schräg in Stücke schneiden. </fr:instrStepText>
        <fr:instrStepText>Die Spargelstücke in etwas Sesamöl rundherum leicht anbraten und salzen. </fr:instrStepText>
      </fr:instruction>
      <fr:instruction>
        <fr:instrStepName>Anrichten</fr:instrStepName>
        <fr:instrStepText>Lachs-Tatar auf die Teller häufeln, Spargelstücke darüber legen. Mit Koriandergrün garnieren. </fr:instrStepText>
      </fr:instruction>
    </fr:recipeInstructions>
    <fr:recipeSideDish>ein Grauer Burgunder vom Hauck Sonnenhof</fr:recipeSideDish>
    <fr:recipeOrigin>Diese Rezept habe ich von Uwe Spitzmüllers Blog '[High Foodality](https://www.highfoodality.de/rezepte/appetizer/lachs-tatar-mit-miso-mirin-marinade-und-gruenem-
spargel/)'.</fr:recipeOrigin>
    <fr:recipeSeeAlso/>
    <fr:recipeLicense/>
</fr:recipe>
```

MD syntax for link
(see cookbook suite)

# 7 kinds of avigation facilities

| | | |
|---|---|---|
|  | Scrolling forward/backward in cookbook (with wrap around) | Tables of content (toc) |
|  | Tabbed chapterwise recipe-level toc pages | |
|  | Chapter-level toc page | |
|  | Recipe-level toc page per chapter (incl. wordcloud) | |
|  | Recipes index | Indices |
|  | Key ingredients index | |
|  | Tiled categories index | |

# Additional material


reachable via
⋯ menu item ("more")
in top-page menu of
cookbooks

# XML text content

Markdown elements contained in XML element text content are converted to corresponding HTML elements

| | |
|---|---|
| **italic** | *italic* |
| __bold__ | **bold** |
| [anchor text](href) | link |

# Design decisions

# File naming conventions

- recipe name
  = image name
  = thumb name

- folder name:
  starts with two decimal digits,
  followed by a blank,
  followed by one or more UTF-8 letters

# Local folder structure

```
…\Websites\kochbuch
│
├──blog_posts          posts for Making of … page
├──categories          different versions of category assignments
├──forms               local form for category assignment
├──fragments           additional information fragments
├──img                 recipes images
├──js                  js files
├──nav                 nav pages (transformation results)
├──recipes             recipe pages and chapterwise recipe-level tocs (transformation results)
├──recipes_xml         cc repo\local
├──sublists_xml        cc repo\local
├──thumbs              thumbs for cat tiles
└──version_history     version notes for version history page
```

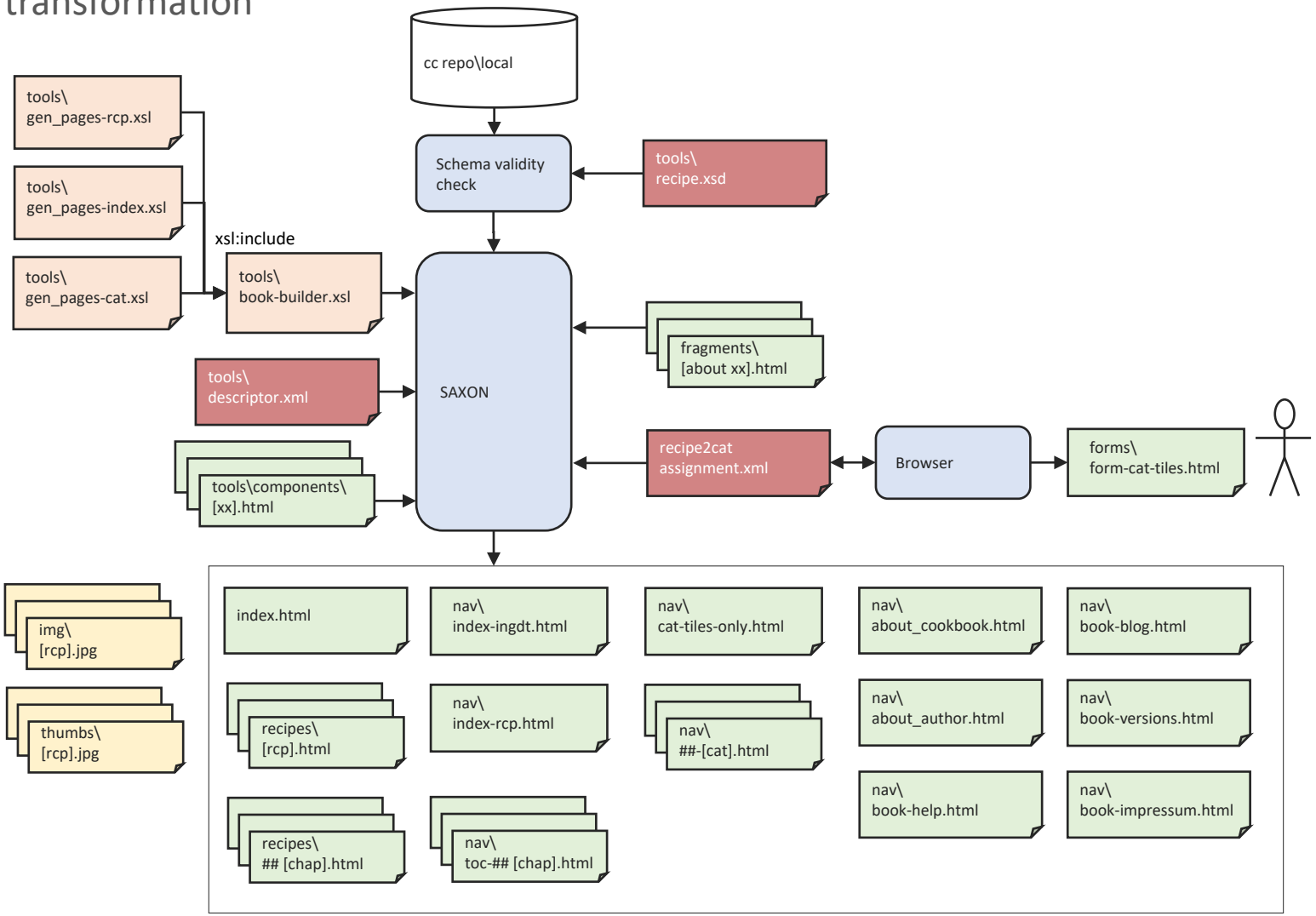
Example: fruschtique – Das Kochbuch

# Customization

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE descriptor SYSTEM "file:///c:/users/nlutt/documents/Websites/tools/entities.dtd">
<fr:descriptor xmlns:fr="http://fruschtique.de/ns/recipe" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://fruschtique.de/ns/recipe recipe.xsd">
  <fr:book>kochbuch</fr:book>
  <fr:fullTitle>fruschtique - Das Kochbuch</fr:fullTitle>
  <fr:path>file:///C:/Users/nlutt/Documents/Websites/kochbuch/</fr:path>
  <fr:logo>img/logos/fruschtique Logo grau transparent.png</fr:logo>
  <fr:about01>
    <fr:headline>Über das Kochbuch</fr:headline>
    <fr:fragment>fragments/html-about_cookbook.html</fr:fragment>
  </fr:about01>
  <fr:about02>
    <fr:headline>Über mich</fr:headline>
    <fr:fragment>fragments/html-about_author.html</fr:fragment>
  </fr:about02>
  <fr:about03>
    <fr:headline>Gebrauchsanweisung</fr:headline>
    <fr:fragment>fragments/html-help.html</fr:fragment>
  </fr:about03>
  <fr:about04>
    <fr:headline>Making of &hellip;</fr:headline>
    <fr:fragment>blog_posts</fr:fragment>
  </fr:about04>
  <fr:about05>
    <fr:headline>Versionsgeschichte</fr:headline>
    <fr:fragment>version_history</fr:fragment>
  </fr:about05>
  <fr:about06>
    <fr:headline>Impressum</fr:headline>
    <fr:fragment>fragments/html-impressum.html</fr:fragment>
  </fr:about06>
</fr:descriptor>
```

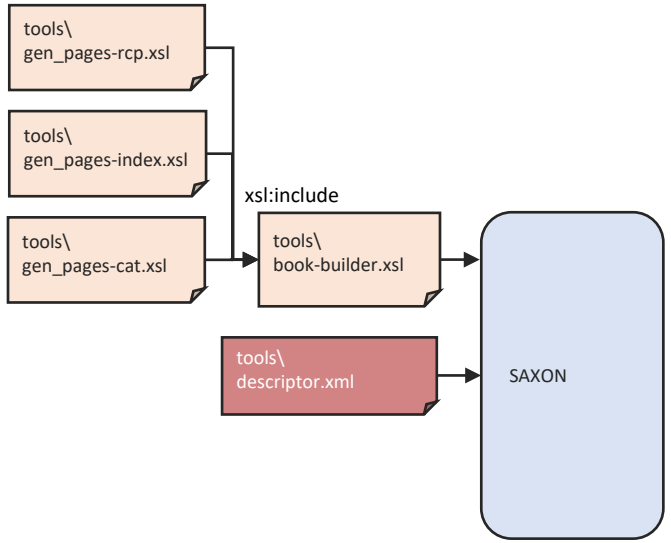
descriptor.xml

# XSLT transformation



# XSLT transformation

- `book-builder.xsl`
  - generate pages for additional information
  - generate index.html
  - apply `gen_pages-xxx.xsl` scripts
- `gen_pages-rcp.xsl`
  - generate recipe pages
  - generate chapterwise recipe-level tocs
    incl. script for wordcloud computation
- `gen_pages-index.xsl`
  - generate tabbed chapter/recipe-level toc page
  - generate recipe index
  - generate key ingredients index
- `gen_pages-cat.xsl`
  - generate page with all category tiles
  - generate pages per category
  - generate form for category assignment
- `descriptor.xml`
  - serves as transformation source doc
  - provides additional parameters
- `SAXON`
  - with oXygen UI



# Nav folder

```
…\Websites\kochbuch
│
⋮
├──nav
│       about_author.html
│       about_cookbook.html
│       about_me.html
│       book-blog.html
│       book-help.html
│       book-impressum.html
│       book-toc.html
│       book-versions.html
│       cat-01-für Gäste.html
│       cat-02-für jeden Tag.html
│       cat-03-für's Büffet.html
│       cat-04-für's Frühstück.html
│       cat-05-zum Sekt.html
│       cat-06-klein, aber fein.html
│       cat-07-Quiches.html
│       cat-08-orientalisch u asiatisch.html
│       cat-09-italienisch.html
│       cat-10-deutsch.html
│       cat-11-meine Lieblinge.html
│       cat-tiles-only.html
│       index-ingdt.html
│       index-rcp.html
│       toc-01 Grundrezepte.html
│       toc-02 Vorspeisen.html
│       toc-03 Suppen.html
│       toc-04 Fleisch.html
│       toc-05 Fisch.html
│       toc-06 Nudeln.html
│       toc-07 Vegetarisch.html
│       toc-08 Beilagen.html
│       toc-09 Salate.html
│       toc-10 Crèmes und Saucen.html
│       toc-11 Süßes.html
│       toc-12 Cocktails.html
⋮
```

additional info pages

Chapter-level toc page

category tiles

Key ingredients index

Recipes index

Tabbed toc pages

Example:
fruschtique – Das Kochbuch

# Upload to Google cloud

For inserting a new recipe:

```
recipes\  before-new.html
recipes\  new.html
recipes\  after-new.html
recipes\  [## chapter].html
img\      new.jpg
thumbs\   new.img
nav\      index-ingdt.html
nav\      index-rcp.html
nav\      toc-[## chapter]
nav\      cat-tiles-only.html
```

For inserting a new recipe including assignment to category:

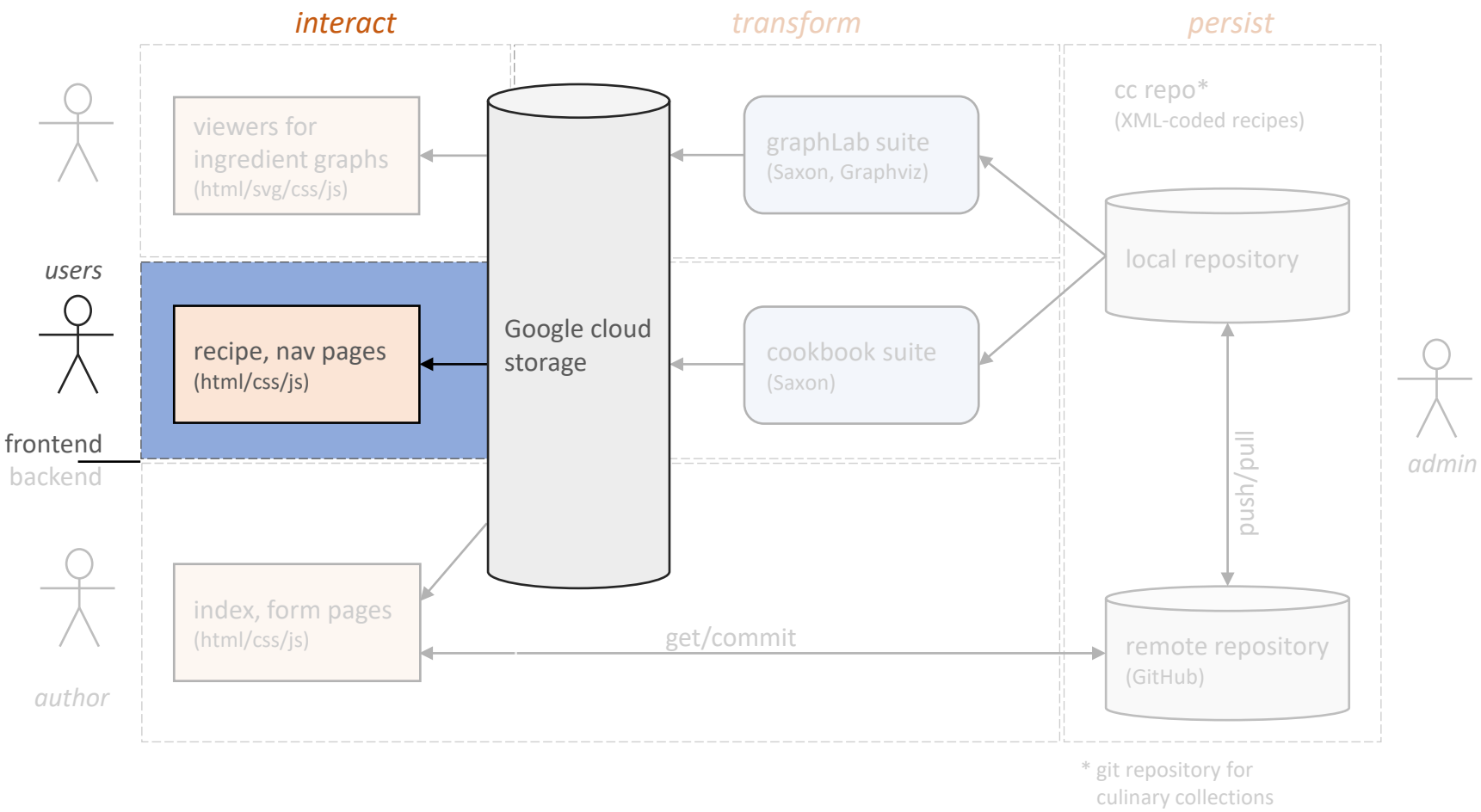
```
recipes\  before-new.html
recipes\  new.html
recipes\  after-new.html
recipes\  [## chapter].html
img\      new.jpg
thumbs\   new.img
nav\      index-ingdt.html
nav\      index-rcp.html
nav\      toc-[## chapter]
nav\      cat-tiles-only.html
nav\      cat-[## cat]
```

For inserting a new image:

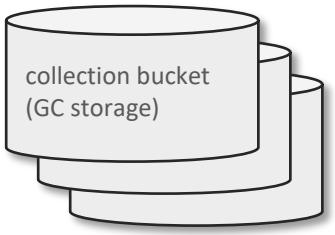
```
img\      new.jpg
thumbs\   new.img
nav\      cat-tiles-only.html
```


Cookbook pages

# fruschtique 3-tier architecture



# GC Storage buckets




1 bucket per collection, e.g.
http://kochbuch.fruschtique.de

- index.html
- recipes
  - recp a.html
  - recp b.html
  - …
- img
  - recp a.jpg
  - recp b.jpg
  - …
- nav
- …

# Design highlights


- Recipe pages layout provides rich image context to recipe texts

- Grid-based layout

fruschtique

# Zucchini-Puffer

Zucchini-Puffer
Nicht sicher | kochbuch.fruschtique.de/recipes/Zucchini-Puffer.html
Pausiert
Apps Advanced Search TV Spielfilm Das Kochbuch fruschtique graphLab fruschtique | ... com... FAZ.NET Google Contacts kopfunter One.com Fritz Fritzchen Weitere Lesezeichen

fruschtique

# Zucchini-Puffer

Zucchini-Puffer kann man warm und kalt essen, als Vor- oder Hauptspeise. Lecker!

Beim Einkauf zu beachten: Eher kleine als große Zucchini kaufen, und niemals die riesigen!

| | |
|---|---|
| 4 | Zucchini |
| 4 Scheiben | Toastbrot |
| 100 g | Kürbiskerne |
| 4 | Eier |
| | Dill |
| | Basilikum getrocknet |
| | Thymian |
| | Oregano |
| | Salbei |
| | Salz, Pfeffer |
| | Öl zum Braten |

**Vorbereitung**

Die Zucchini streifig schälen, dann grob raspeln, salzen und die Raspel für ca. ½ h stehen lassen, damit sie Wasser ziehen. Das Toastbrot würfeln. Die Kürbiskerne leicht anrösten und dann schroten.

**Pufferteig**

Das Toastbrot in kleine Würfel schneiden. Die Kürbiskerne in einem Blitzhacker zerkleinern; das Ergebnis soll fast so fein sein wie ein Weizenschrot.

Die Zucchini-Raspel fest ausdrücken; das gelingt am besten mit einer Spätzlespresse. Raspel, Brotstücke, Kürbiskernschrot, Kräuter und Eier miteinander vermischen. Mit Salz und Pfeffer abschmecken.

**Braten**

Backofen auf 120 °C vorheizen. Ein Backblech mit Backpapier belegen.

Nun kommt das schwierigste: das Braten. Öl in einer Pfanne mäßig erhitzen. Brät man den Pufferteig zu scharf an, wird er schwarz. Brät man ihn nicht heiß genug an, wird das Ei nicht richtig fest. Erfahrung ist gefragt. Man kann aus dem Teig Puffer machen, die so groß wie die Pfanne sind, oder man kann kleine Puffer machen, z.B. in einem Edelstahlring. Nachdem man den Teig in die Pfanne gegeben hat, sollte man die Ränder gut glätten.

Nach einer Bratzeit von 5 min auf jeder Seite sollten die Puffer eine schöne Farbe angenommen haben und fest geworden sein. Die zweite Seite braucht nicht ganz soviel Zeit wie die erste.

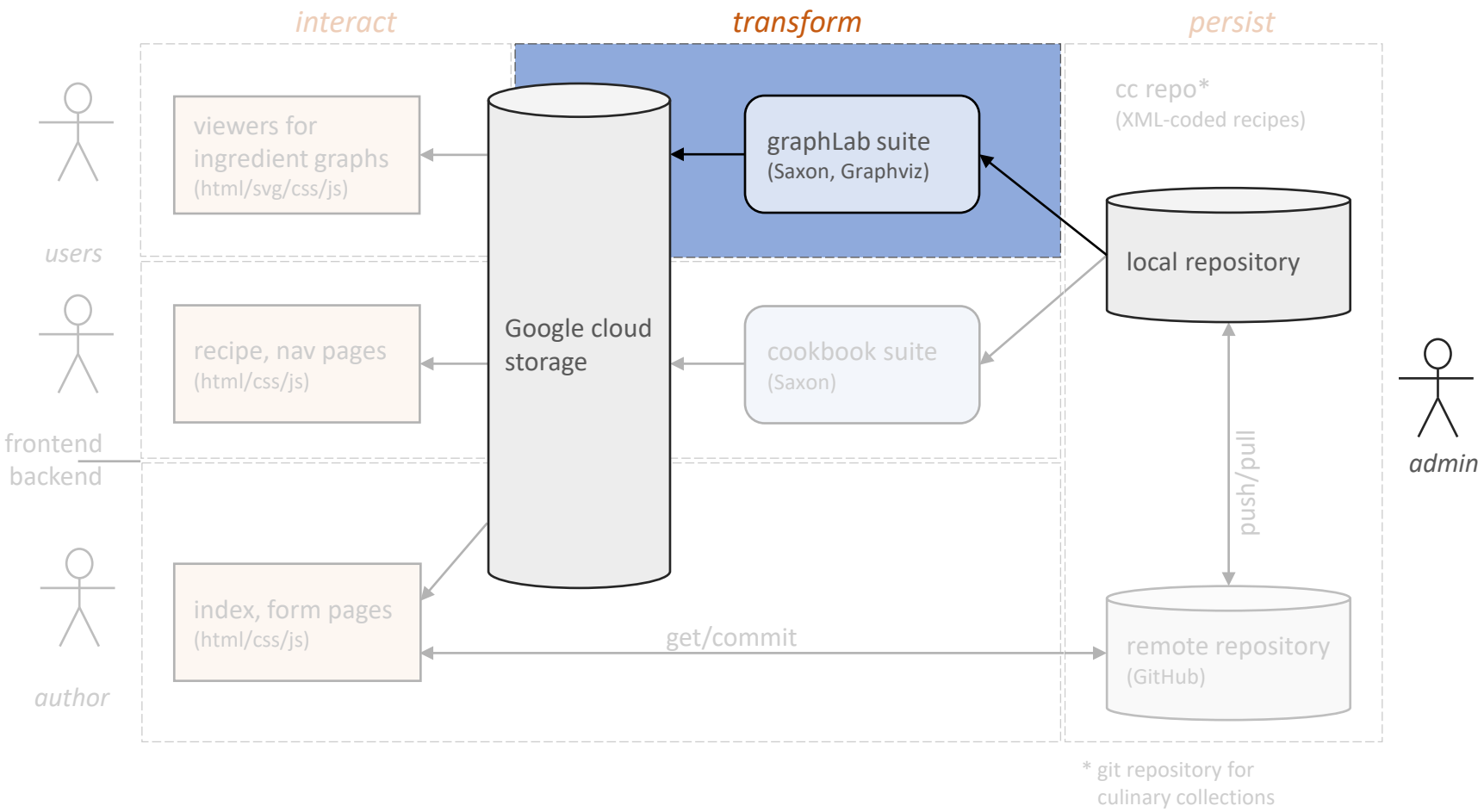
Die gebratenen Puffer in den Backofen verfrachten und noch etwas garen lassen.

Dazu:

Eine Joghurt-Dill-Sauce und ein leichter Rotwein (Du weißt schon: der Rote, in dem die kleinen Sommermücken so gerne baden ...)


GraphLab suite

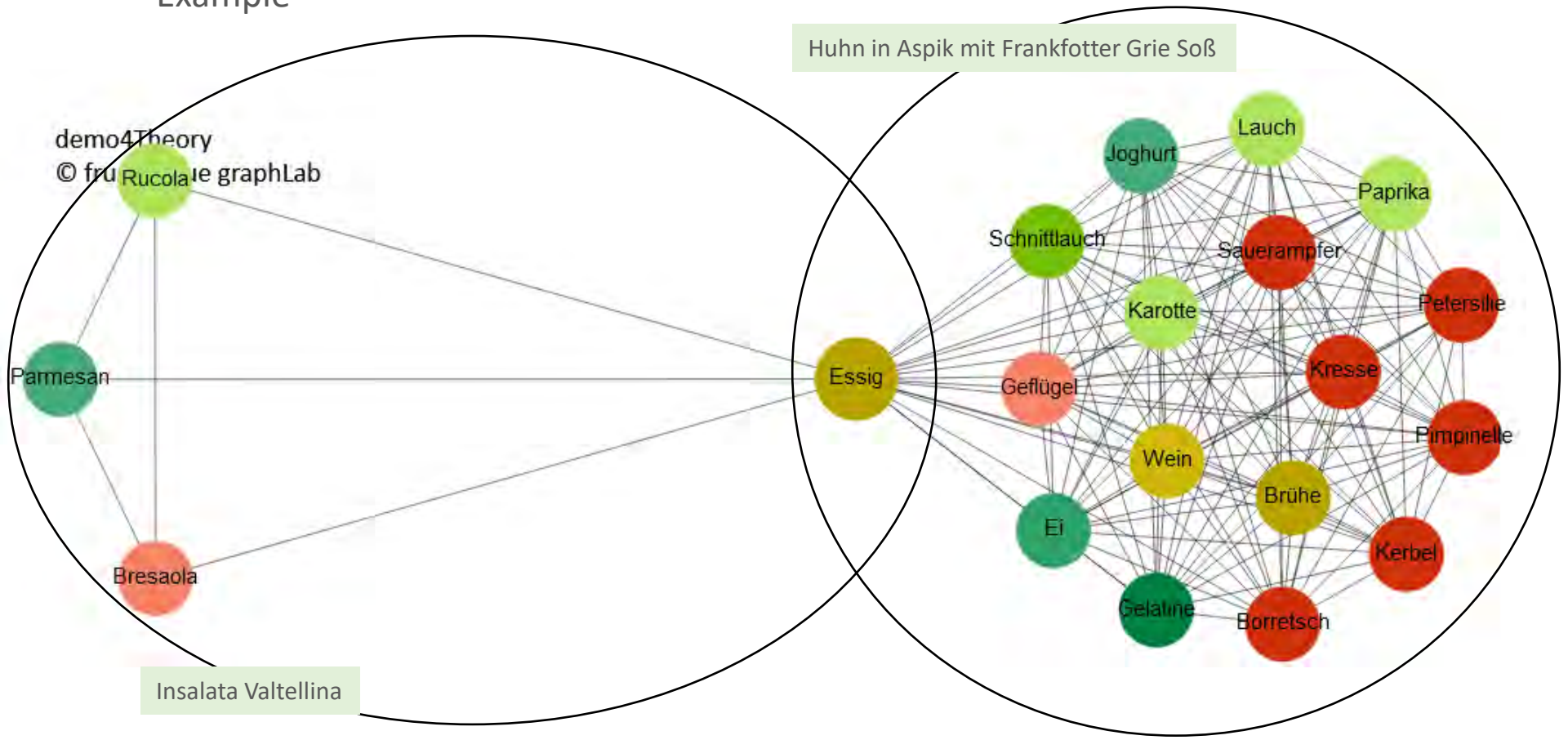
# fruschtique 3-tier architecture



## Concept

- Select any number of recipes from any number of fruschtique collections; call it: *selection*
- Model a selection as a social graph:
  - The set of pairwise distinct ingredients occuring in a selection acts as node set of the graph.
  - A relation between any two ingredients exists, if these two ingredients occur together in at least one recipe.
  - The full social graph is composed from so-called recipe graphs (see next slide).
  - Call it: *ingredient graph*.
- Provide to the user a GUI for *exploring ingredient graphs*.

# Example




node color: inredient class
node diameter: ingredient prevalence

- Each recipe is modelled as complete graph, call it: ***recipe graph***
- Shared ingredients (here: "Essig") interconnect recipe graphs.

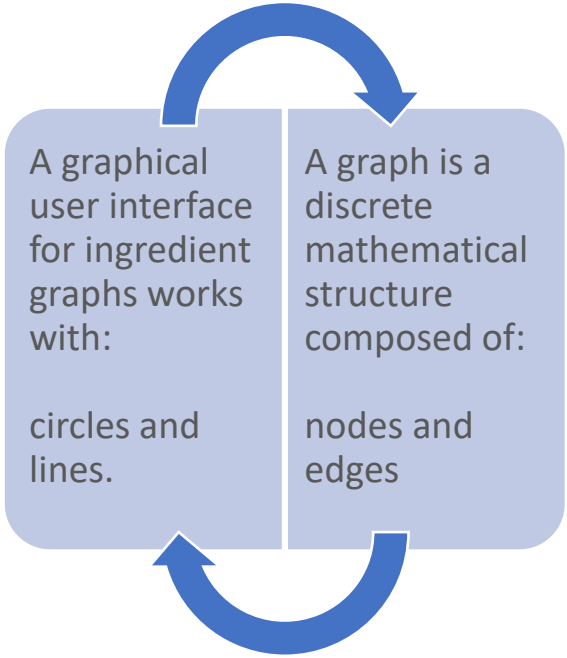
# Graph-theoretic interlude

An ingredient graph $IG = (V, E, w, c, p)$ is an undirected, loop-free, weighted, and node-labeled graph where

- $V$ is a finite set of ingredient nodes identified by their ID and
- $E \subseteq \binom{V}{2}$ is a set of edges.
- $w$ is a function that assigns a weight value from $\mathbb{N}\backslash\{0\}$ to each edge,
- $c$ is a function that assigns a class value to each node, where this value comes from $class = \{meat, fish, herb, spice, veg, fruit, onion, nuts, carb, sweet, alc, condi, milk, egg, fat, etc\}$
- $p$ is a function that assigns a prevalence value $\varphi$ to each node with $\varphi \in \mathbb{Q} \wedge \varphi > 0$.

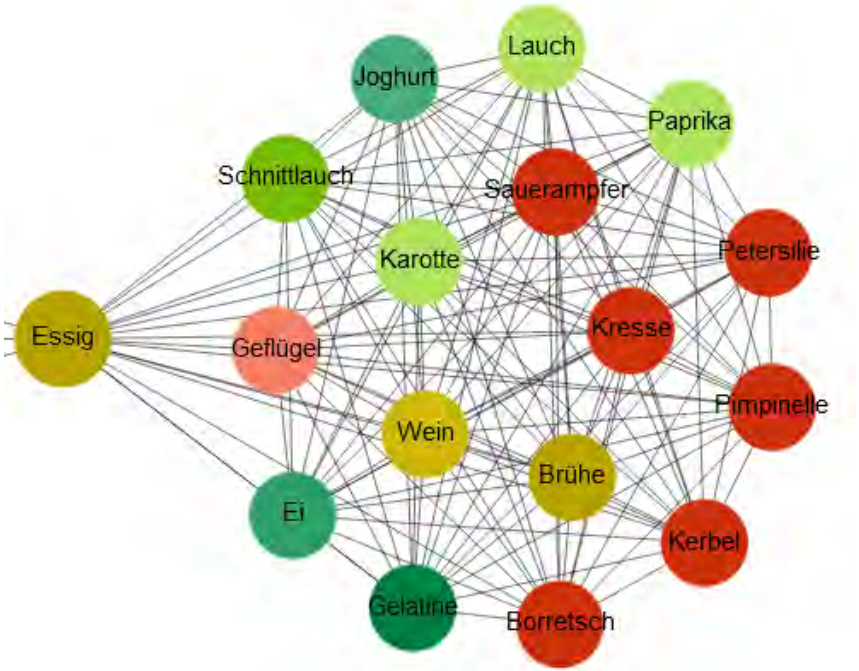
# A note on terminology:

Graphs are not graphics.



## *La trahison des images*

René Magritte


*This is not a graph.*

# Design decisions

- Provide web pages for
  different collections of recipes via:
  `viewer-collection.html?coll=[xx]`
- When viewer is opened,
  load the related graph, graphics, and
  specific information from
  Google cloud storage

# graphLab suite: processing overview

4 core steps

- Selection
  - select recipes for graph computation
  - XSLT script
- "poorGraph"
  - extract node set and edge multiset from recipes in selection
  - XSLT script
- Layout computation
  - result in SVG format
- "richGraph"
  - add additional information to layout
  - code graph and graphics for GC storage and browser fetch
  - XSLT script

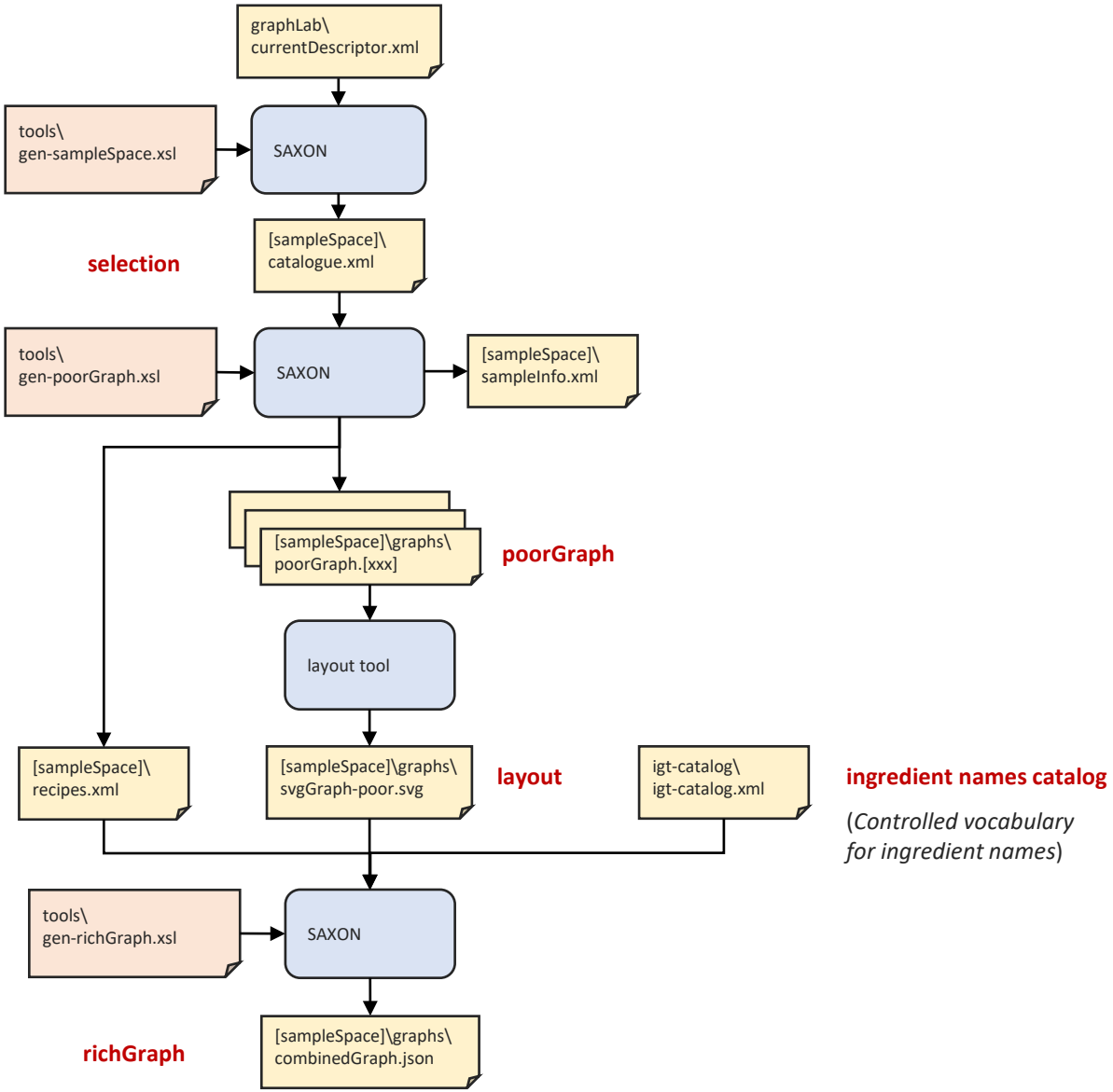
Saxon (<oXygen/> XML editor)

Saxon

Graphviz
(Graph Visualization Software)

Saxon

# graphLab suite: step-by-step



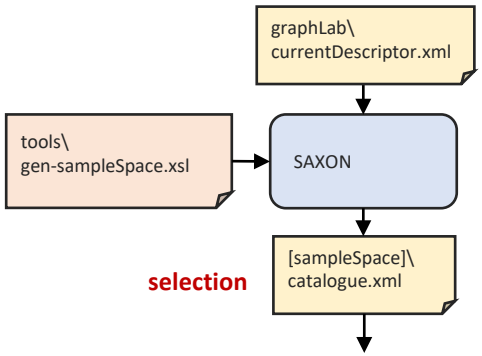
# graphLab descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<fe:experiment xmlns:fe="http://fruschtique.de/ns/fe" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://fruschtique.de/ns/fe
file:///c:/users/nlutt/documents/websites/graphlab/tools/experiment.xsd">
  <fe:fullTitle>ckMaronensuppe</fe:fullTitle>
  <!-- path to recipes_xml folder in cookbook folder -->
  <fe:cookbook>C:/Users/nlutt/Documents/Websites/ckCollections/recipes_xml</fe:cookbook>
  <fe:experimentPath>sampleSpaces/ckMaronensuppe/</fe:experimentPath>
  <fe:win-experimentPath>sampleSpaces\ckMaronensuppe\</fe:win-experimentPath>
  <fe:experimentName>Maronensuppe</fe:experimentName>
  <fe:experimentDescription>Rezepte zum Stichwort Maronensuppe von chefkoch.de</fe:experimentDescription>
</fe:experiment>
```

currentDescriptor ckMaronensuppe.xml

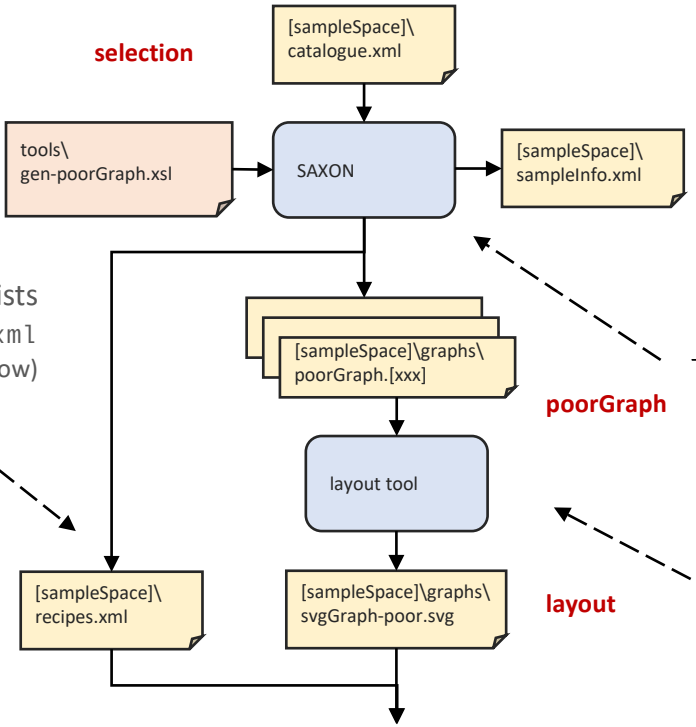
Note: Here, you may read "experiment" as "selection".

# graphLab suite: selection




- Create required directories in `graphLab\sampleSpaces`
- Create command line for graphics layout tool
- Select recipes and build selection catalogue `(sampleSpaces\[selection]\catalogue.xml)`

# graphLab suite: "poorGraph" and layout




Create `sampleInfo.xml` (see below)

Store ingredients lists recipe-wise in `recipes.xml` (see below)

**poorGraph**

- Create node set (incl. ingredient occurence count) and edge multiset
- Derive weighted graph by *collect-and-reduce* algorithm
- Textual graph descriptions in different formats

**layout**

Produce layout for weighted graph in SVG format

# Collect edges

```
<xsl:for-each select="$recipeIgt/igt">
  <xsl:variable name="current" select="."/>
  <xsl:for-each select="following-sibling::node()">
    <!-- eliminate self loops -->
    <xsl:if test="not($current = .)">
      <collectedEdge>
        <xsl:choose>
          <xsl:when test="compare($current, ., 'http://www.w3.org/2013/collation/UCA?lang=de;strength=primary') = 1">
            <source>
              <xsl:value-of select="."/>
            </source>
            <target>
              <xsl:value-of select="$current"/>
            </target>
          </xsl:when>
          <xsl:otherwise>
            <source>
              <xsl:value-of select="$current"/>
            </source>
            <target>
              <xsl:value-of select="."/>
            </target>
          </xsl:otherwise>
        </xsl:choose>
      </collectedEdge>
    </xsl:if>
  </xsl:for-each>
</xsl:for-each>
```

gen_poorGraph.xsl

# Identifying edges

In most textual graph descriptions, edges are identified by a couple of node IDs. In directed graphs, these IDs may be read as "source" and "target".

Obviously, in undirected graphs, node ID couples `(1,2)` and `(2,1)` denote the same edge.

Thus, when converting an undirected multigraph into an undirected weighted graph, edges `(1,2)` and `(2,1)` must be combined to a single edge with attribute `weight=2`.

To ease graph processing, edges in fruschtique graphs carry an ID attribute `[node1]—[node2]`, where `node1` $<_{\texttt{by\_collation}}$ `node2`.

# A note on sorting

In fruschtique, computations on edges are performed both in XSLT scripts and in JS functions.

To form same edge IDs, the same collations must be used in both languages.

In XSLT, the collation is

```
'http://www.w3.org/2013/collation/UCA?lang=de;
strength=primary'
```

In JS, the collation is

```
a.localeCompare(b, 'de-DE-1996')
```

# Convert multigraph to weighted graph

```
<edges>
  <!-- eliminate duplicate edges and insert edge weight instead -->
  <xsl:for-each-group select="$edgeCollector/collectedEdge" group-by=".">
    <xsl:sort select="." collation="http://www.w3.org/2013/collation/UCA?lang=de;strength=primary"/>
    <edge>
      <xsl:attribute name="id">
        <xsl:value-of select="concat(source, '--', target)"/>
      </xsl:attribute>
      <source>
        <xsl:value-of select="source"/>
      </source>
      <target>
        <xsl:value-of select="target"/>
      </target>
      <weight>
        <xsl:value-of select="count(current-group())"/>
      </weight>
    </edge>
  </xsl:for-each-group>
</edges>
```

gen_poorGraph.xsl

# Graph in .dot format (snippet)

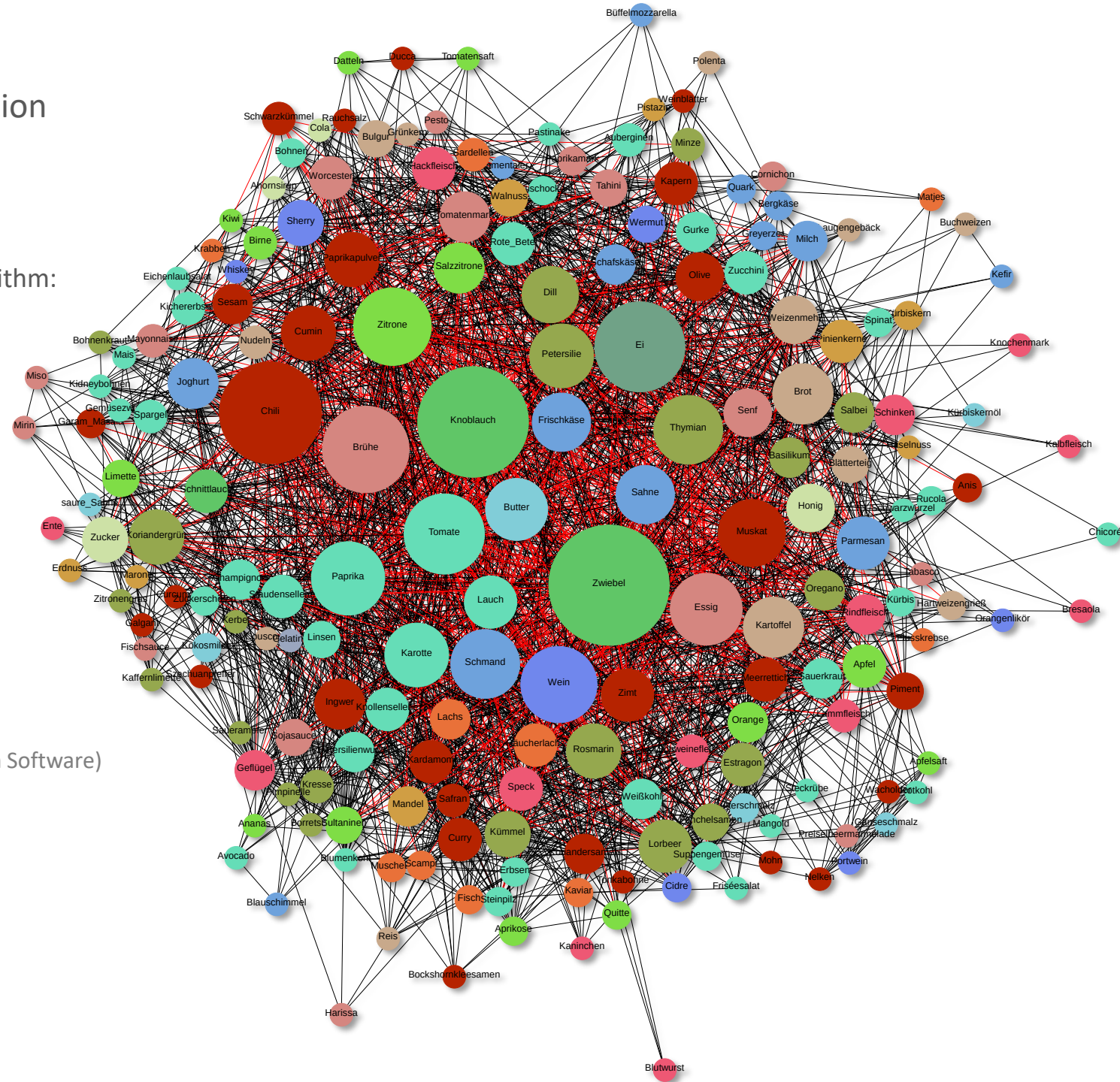
.dot format is input for generating SVG graphics

- "width" attribute reflects ingredient prevalence
- "penwidth" and "color" attributes reflect edge weight

```
apfel [width=5.5, label=Apfel]
apfelkraut [width=2.5, label=Apfelkraut]
austern [width=2.5, label=Austern]
basilikum [width=4.674234614174766, label=Basilikum]
bier [width=3.598076211353316, label=Bier]
birne [width=4, label=Birne]
birnenkraut [width=4.354101966249685, label=Birnenkraut]
blätterteig [width=2.5, label=Blätterteig]
…
bier--stärke [penwidth=1]
bier--weinraute [penwidth=1]
bier -- weizenmehl [penwidth=1]
bier--zitrone [penwidth=1]
bier--zwiebel [penwidth=3, color=Red]
birne--bratenfond [penwidth=1]
birne--bratfett [penwidth=1]
birne--brot [penwidth=1]
birne--butter [penwidth=3, color=Red]
birne--essig [penwidth=4, color=Red]
birne--estragon [penwidth=1]
birne--herz [penwidth=1]
birne--kalbskopf [penwidth=1]
birne--karotte [penwidth=1]
```

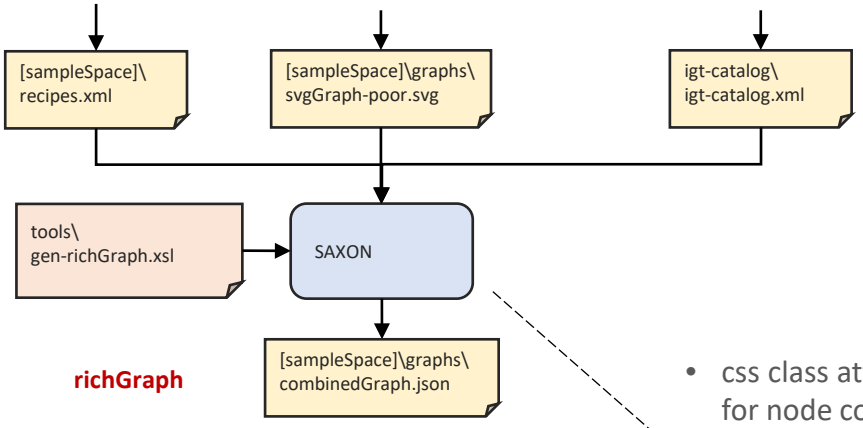
# Layout creation

- sfdp layout algorithm: *"multi-scale, force-directed approach based on spring model"*
- output: SVG




Graphviz
(Graph Visualization Software)

# graphLab suite: "richGraph"




- css class attributes
  for node coloring
- transfer coding
  for graph and graphics

# Transfer coding for graph and graphics

Design choices

- graph
  - textual description

  or

  - compute graph in browser from recipe-wise ingredients lists
- graphics
  - compute layout in browser from textual description

  or

  - transfer static layout in SVG format

# Design decisions

- Graph: transfer recipe-wise ingredients lists and compute graph in browser
- Graphics: transfer static layout in SVG format
- For transfer, combine both in single JSON file: `combinedGraph.json`

## Pros

- recipe information available in browser
- ingredients lists size < textual descr. size
- ready-made SVG can easily be integrated in HTML page

## Cons

- SVG possibly yields large file size
- graph computation in browser may take considerable amount of time

# Converting XML → JSON

XPath and XQuery Functions and Operators 3.1
W3C Recommendation 21 March 2017, Chap 17.4

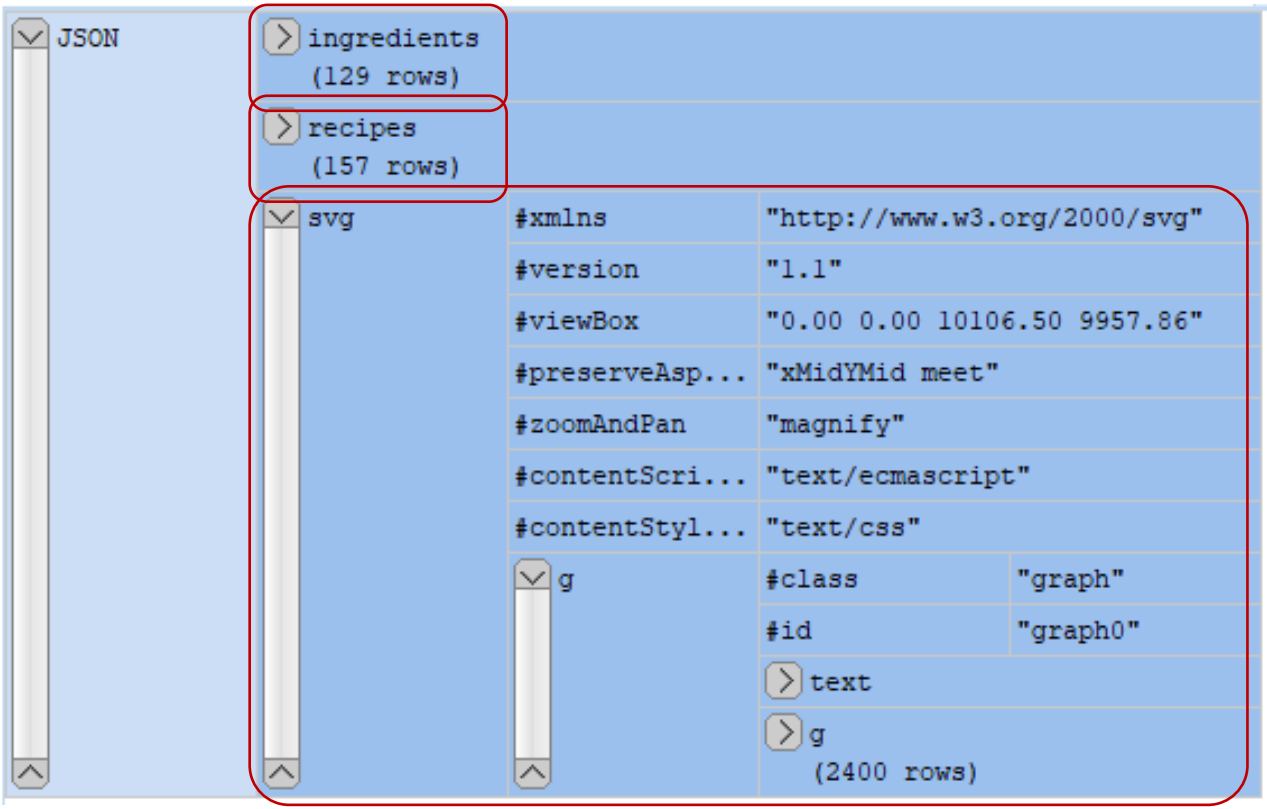
```
<xsl:template match="/">
    <!-- Further templates are applied to svg file as declared in descriptor -->
    <xsl:variable name="xx">
      <map xmlns="http://www.w3.org/2005/xpath-functions">
        <xsl:apply-templates select="$recipes" mode="recipes"/>
        <map key="svg">
          <xsl:apply-templates select="$svg" mode="svgGraph"/>
        </map>
      </map>
    </xsl:variable>
    <xsl:result-document href="{$path2combined}">
      <xsl:sequence select="xml-to-json($xx)"/>
    </xsl:result-document>
    <xsl:variable name="src">
      <xsl:value-of select="$path2combined"/>
    </xsl:variable>
    <xsl:variable name="tgt">
      <xsl:value-of select="concat($path,'dist/g_',$experimentName,'/combinedGraph.json')"/>
    </xsl:variable>
    <xsl:sequence select="fs:copy($src, $tgt)"/>
  </xsl:template>

  <xsl:template match="ingredients" mode="recipes">
    <array key="ingredients">
      <xsl:for-each select="ingredient">
        <map>
          <string key="id">
            <xsl:value-of select="@id"/>
          </string>
          <string key="label">
            <xsl:value-of select="label"/>
          </string>
          <string key="occurs">
            <xsl:value-of select="occurs"/>
          </string>
        </map>
      </xsl:for-each>
    </array>
  </xsl:template>

  <xsl:template match="recipes" mode="recipes">
    <array key="recipes">
      <xsl:for-each select="recipe">
        <map>
          <string key="recipeName">
            <xsl:value-of select="name"/>
          </string>
          <array key="ingredients">
            <xsl:for-each select="ingredients/ingredient">
              <string>
                <xsl:value-of select="."/>
              </string>
            </xsl:for-each>
          </array>
        </map>
      </xsl:for-each>
    </array>
  </xsl:template>
```

# combinedGraph.json (example)

- Stored in
  GC storage bucket
- Fetched
  by browser
  on demand



# combinedGraph.json

Snippet from the ingredients section, listing all ingredients used in selection


```
{
  "ingredients": [
    {
      "id": "ahornsirup",
      "label": "Ahornsirup",
      "occurs": "1"
    },
    {
      "id": "ananas",
      "label": "Ananas",
      "occurs": "1"
    },
    {
      "id": "anis",
      "label": "Anis",
      "occurs": "2"
    },
    {
      "id": "apfel",
      "label": "Apfel",
      "occurs": "6"
    },
    {
      "id": "apfelsaft",
      "label": "Apfelsaft",
      "occurs": "1"
    },
    {
      "id": "aprikose",
      "label": "Aprikose",
      "occurs": "2"
    },
    {  ...
```

# combinedGraph.json

Snippet from the recipes section, listing all ingredients lists


```
"recipes": [
    {
      "recipeName": "Schweinefilet im Blätterteig",
      "ingredients": [
        "schweinefleisch",
        "piment",
        "rosmarin",
        "senf",
        "petersilie",
        "thymian",
        "brot",
        "haselnuss",
        "knoblauch",
        "zitrone",
        "chili",
        "blätterteig",
        "schinken",
        "ei",
        "milch"
      ]
    },
    {
      "recipeName": "Kürbiskernpesto",
      "ingredients": [
        "kürbiskern",
        "parmesan",
        "zitrone",
        "kürbiskernöl",
        "petersilie",
        "brühe"
      ]
    },
    { ...
```

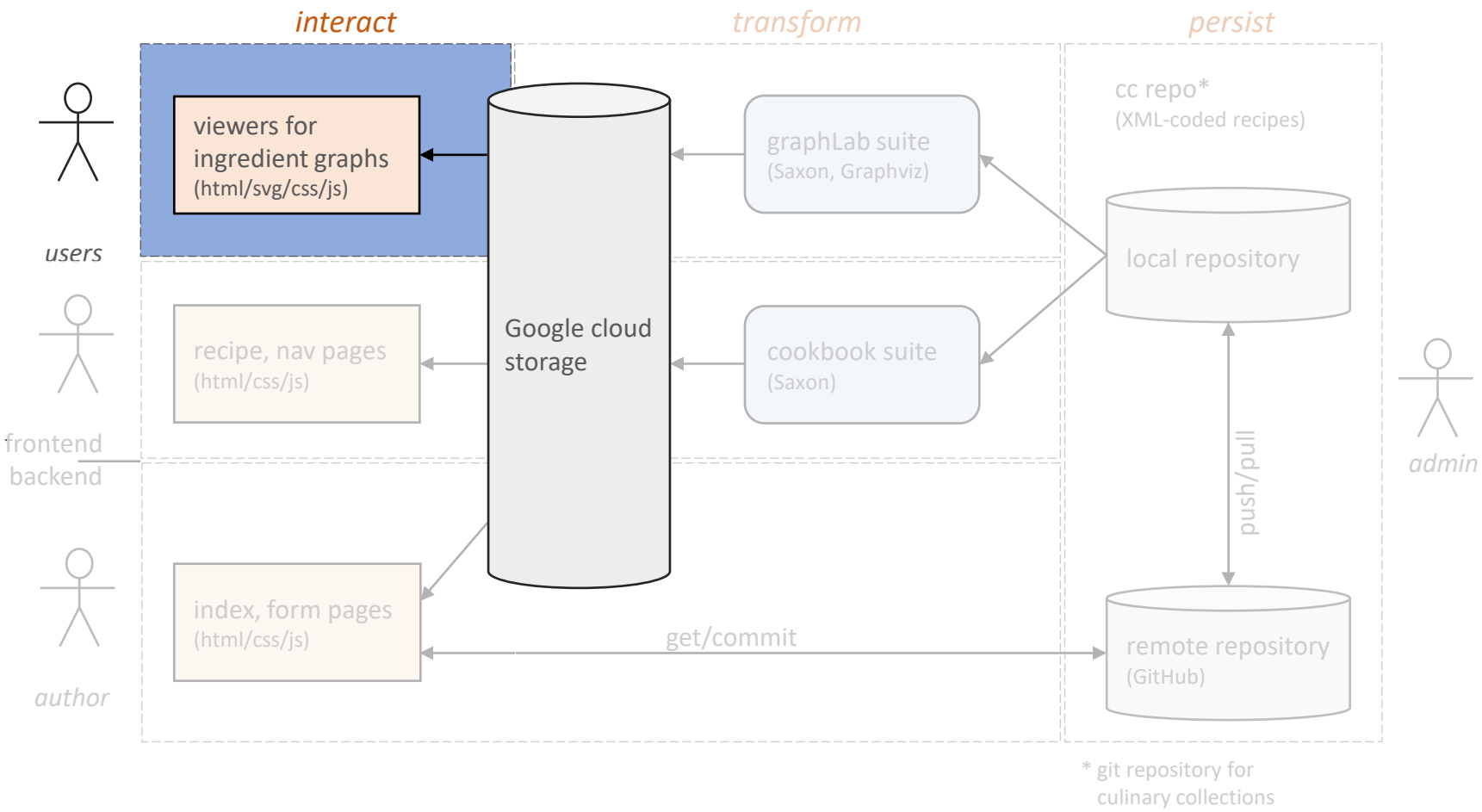
# combinedGraph.json

Snippet from the SVG section, including SVG header


```
"svg": {
    "#xmlns": "http://www.w3.org/2000/svg",
    "#version": "1.1",
    "#viewBox": "0.00 0.00 8370.18 8132.01",
    "#preserveAspectRatio": "xMidYMid meet",
    "#zoomAndPan": "magnify",
    "#contentScriptType": "text/ecmascript",
    "#contentStyleType": "text/css",
    "g": {
      "#class": "graph",
      "#id": "graph0",
      "text": {
        "#class": "g-text",
        "#x": "68",
        "tspan": [
          {
            "#y": "100",
            "#text-content": "all-2"
          },
          {
            "#x": "68",
            "#dy": "1.2em",
            "#text-content": "© fruschtique graphLab"
          }
        ]
      },
      "g": [
        {
          "#class": "edge",
          "#id": "ahornsirup--chili",
          "path": {
            "#fill": "none",
            "#stroke": "black",
            "#d": "M7053.81,3667.79L2937.34,3770.42"
          }
        },
        {
          "#class": "edge",
          "#id": "ahornsirup--cola",
          "path": {
            "#fill": "none",
            "#stroke": "black",
            "#d": "M7080.52,3600.98L6162.99,2665.88"
          }
        },
        { ...
```


GraphLab pages

# fruschtique 3-tier architecture



# Processing concepts

- JSON-to-SVG parser
- In-browser graph creation
- "Connected Components" function for graph analysis
- Lorenz curve and Gini coefficient functions for collection analysis

# In-browser processing APIs


generating and exploring graphs


generating and editing SVG graphics


generating statistics diagrams

simple statistics

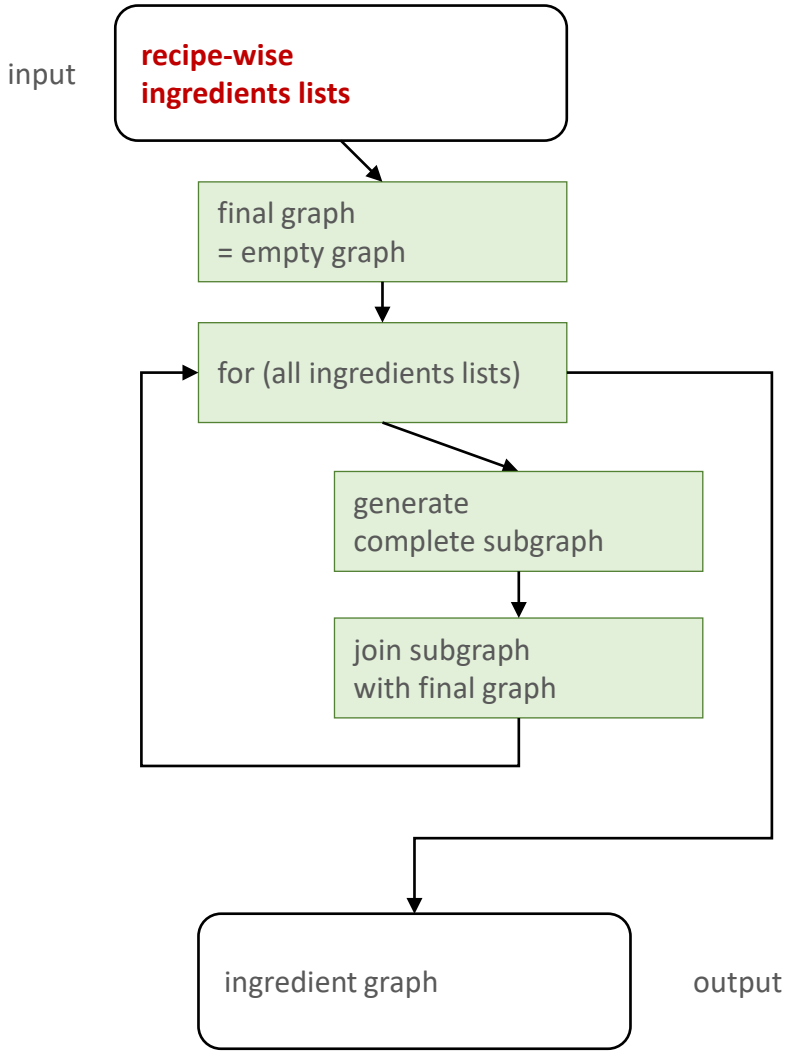
# JSON-to-SVG parser

- Processing of nested structures supported by 2 "in-sync" stacks ("keller"):
  - `kellerObj` (for objects)
  - `kellerPar` (for parents)
- Arrays are processed iteratively

```
function svgGraphBuilder(g) {
 svgGraphs.forEach((graph) => {
  let parent = graph; // svg graph instance
  let child = parent;
  let element = g.svg; // svg element in json file
  let kellerObj =[];
  let kellerPar =[];
  kellerObj.push(element);
  kellerPar.push(parent);
  while ((element = kellerObj.pop()) !== undefined) {
   parent = kellerPar.pop();
   for (let[key, value] of Object.entries(element)) {
    if ((typeof (value) === "object") && (Array.isArray(value) === false)) {
     child = graph.el(key);
     parent.append(child);
     kellerObj.push(value);
     kellerPar.push(child);
    } else if ((typeof (value) !== "object") && (Array.isArray(value) === false)) {
     if (key === "#text-content") {
      parent.node.innerHTML = value;
     } else if (key === "title") {
      child = graph.el(key);
      parent.append(child);
      child.node.innerHTML = value;
      } else {
       let att = key.substr((key.indexOf('#') + 1), key.length);
       parent.attr({[att]: value});
      }
    }
    if (Array.isArray(value) === true) {
     for (let[k, v] of Object.entries(value)) {
      child = graph.el(key);
      parent.append(child);
      kellerObj.push(v);
      kellerPar.push(child);
     }
    }
   }
  }
 })
}
```

showMaster.js

# In-browser graph creation




See details here:

**JSNetworkX**

A JavaScript port of the NetworkX graph library.

# "Connected Components" function for graph analysis

- Not part of JSNetworkX
- Implementation follows well-known algorithm, e.g. here

```
// compute connected components
function connComp() {
 // init
 function init() {
  for (let nd of G.nodes()) {
   // node attributes object
   let n = G.node.get (nd);
   // add connected component attribute
   n.cc = 0;
  }
 }
 // recursive depth first search
 function dfs(nd, countCC) {
  let n = G.node.get (nd);
  // node attributes object
  n.cc = countCC;
  for (let neighbor of jsnx.neighbors(G, nd)) {
   let n = G.node.get (neighbor);
   if (n.cc === 0) {
    n.cc = countCC;
    dfs(neighbor, countCC);
   }
  }
 }
 // main
 init();
 let countCC = 0;
 for (let nd of G.nodes()) {
  let n = G.node. get (nd);
  if (n.cc === 0) {
   countCC++;
   dfs(nd, countCC);
  }
 }
 let nodesCC = G.nodes(true);
 let cc = [];
 // result array of arrays
 for (let i = 1; i <= countCC; i++) {
  // loop over connected components
  let temp = [];
  for (let j = 0; j < nodesCC.length; j++) {
   // loop over source array
   if (nodesCC[j][1].cc === i) {
    temp.push(nodesCC[j]);
   }
  }
  cc[i-1] = temp;
 }
 return (cc.length);
}
```

showMaster.js

# Lorenz curve function

array of [ID, value] couples

simple statistics API

```
// compute Lorenz curve
function Lorenz(arr) {
 let step = 0;
 let yy = [];
 arr.forEach(function (el, i) {
  yy[i] = el[1] + step;
  step = el[1] + step;
 });
 let xVal = [];
 let yVal = [];
 xVal[0] = yVal[0] = 0;
 for (let i=1;i<10;i++) {
  xVal[i] = i*10;
  yVal[i] = ss.quantile (yy, i*0.1) / yy[yy.length-1] * 100;
 }
 xVal[10] = yVal[10] = 100;
 return {x: xVal, y: yVal}
}
```

reference as
`let x = Lorenz(arr).x;`

# Gini coefficient function

array of [ID, value] couples

```
// compute Gini coefficient
function Gini(curve) {
  let belowLorenz = 0, prev = 0, total = 0;
  curve.forEach(function (el) {
    belowLorenz += el[1] / 2 + prev;
    prev += el[1];
    total += el[1];
  })
  let triangle = total * (curve.length) / 2;
  let GC = (triangle - belowLorenz) / triangle;
  return (GC);
}
```

# Design highlights

Card-based top-level menu

Structured left-side menu for graph exploration

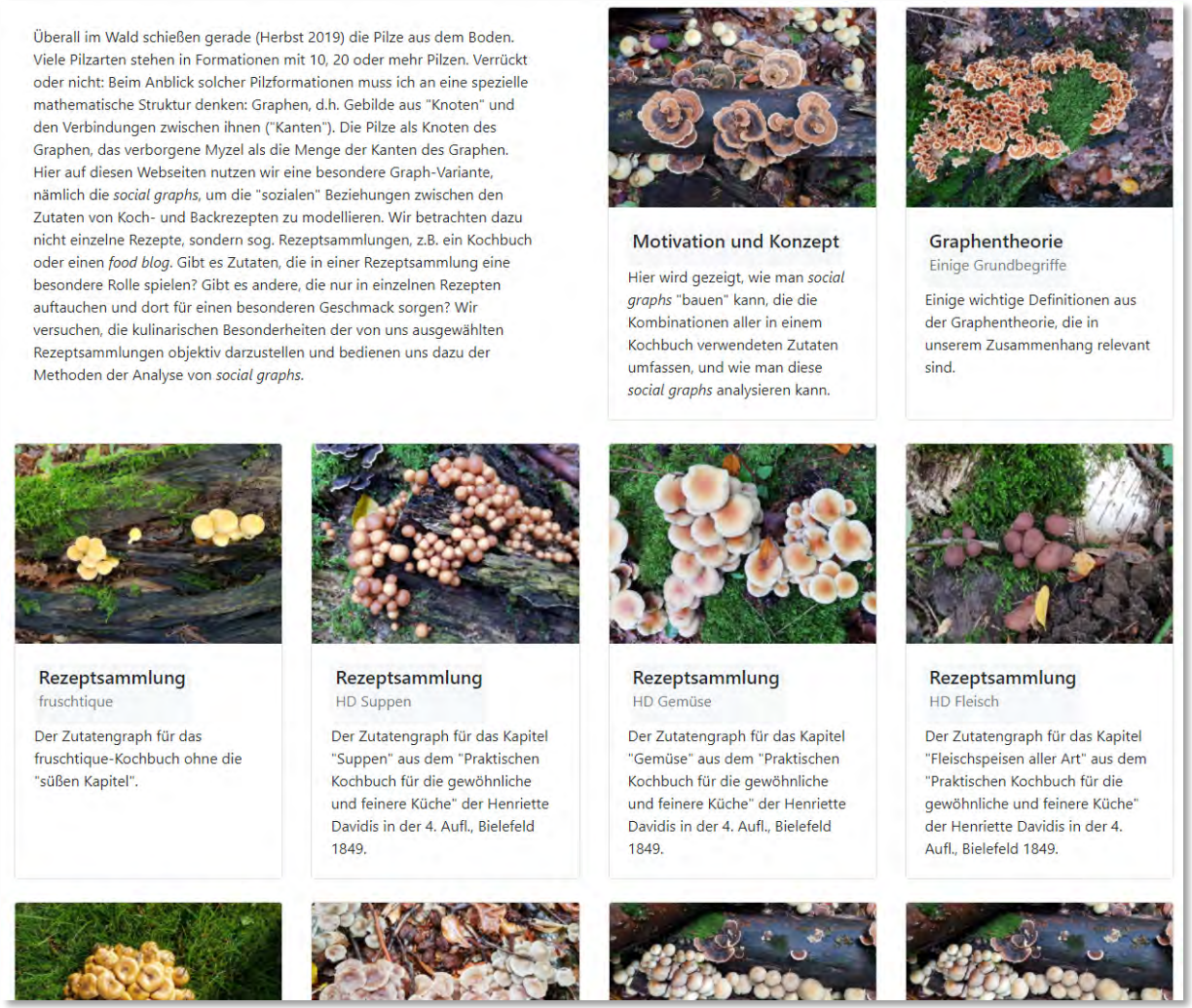

Fog over graph

Rich image context to statistics charts

Bootstrap 12 column layout

# Card-based top-level menu

- top-level navigation in graph viewer
- easily extensible
- mycelia images as "models" for graphs
- one dedicated viewer page assigned to each card
- target for back arrow in viewer pages

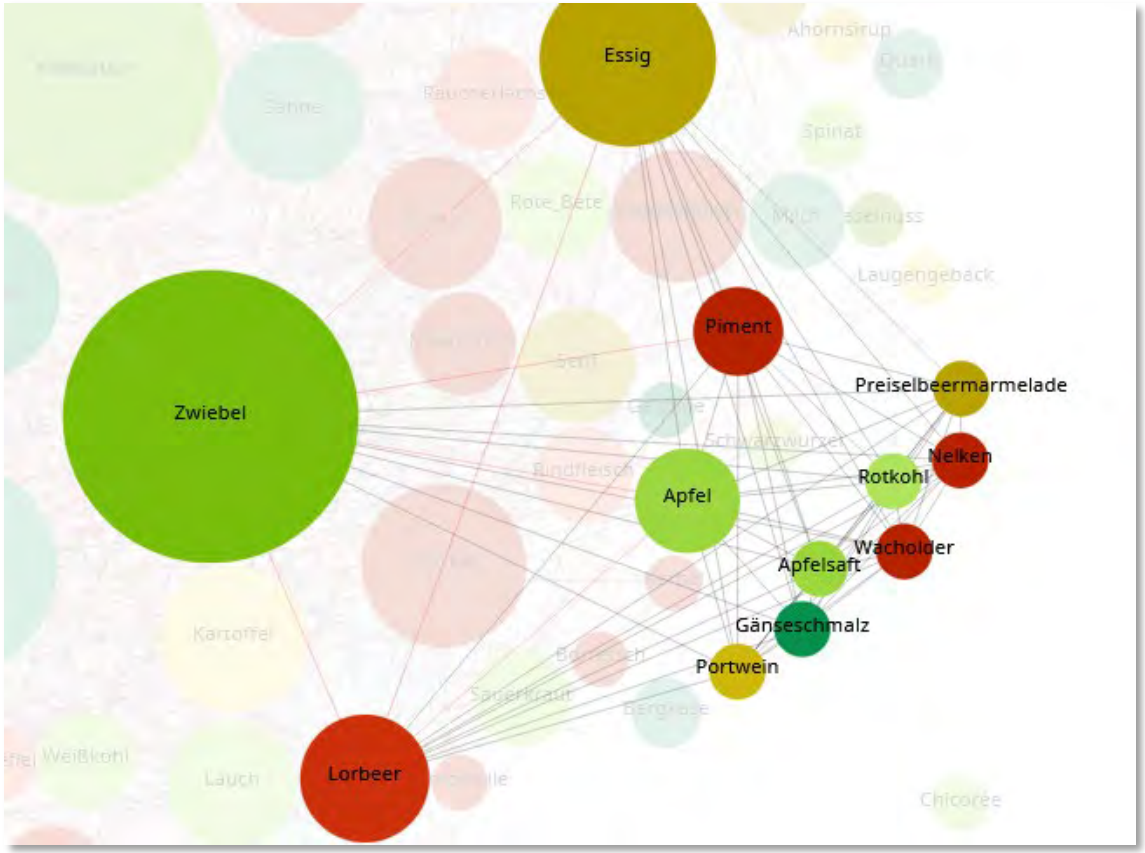


# Structured left-side menu for graph exploration

- navigation element for graph pages
- easily extensible
- implemented as Bootstrap accordion
- back arrow points to top-level menu

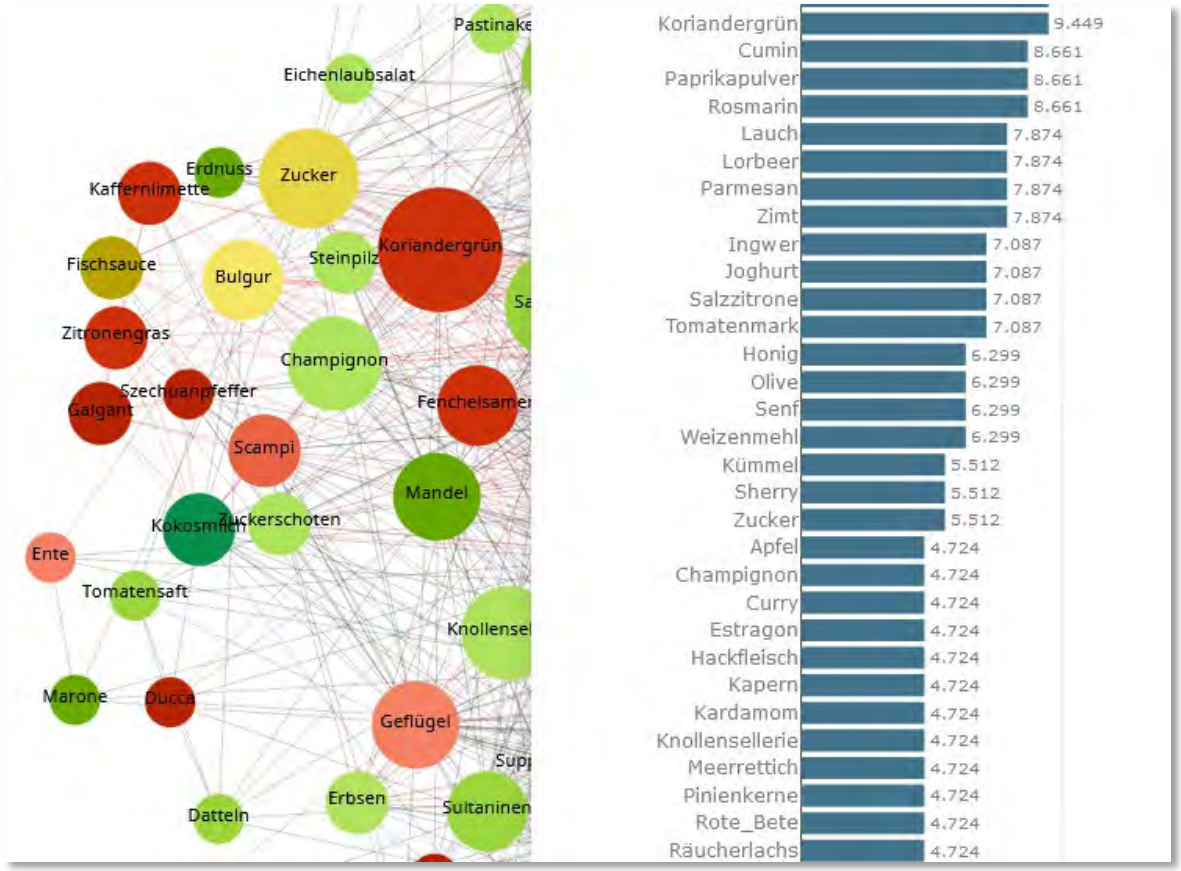
# Fog over graph

- some views on graph require some elements to stand out
  - → let all other elements by covered by "fog"
  - → all elements' positions are kept
- fog implemented as semi-transparent white rectangular plane laid over graph
- elements to stand out are cloned and attached to fog plane
- when fog is removed, all attached elements are removed together with fog



# Rich image context to statistics charts

- Fly-in charts
- Z-index control



# Bootstrap 12 column layout

single Bootstrap row

3 columns accordion navigation

9 columns SVG graph space

out-of-flow chart spaces

autocomplete info

scripts

```
<body onload="fetchGraph('g_k60')">
  <div id="main">
    <!-- ------------------------------------------------------ //-->
    <!-- single row                                             //-->
    <!-- ------------------------------------------------------ //-->
    <div class="row mt-2 row-height">
      <!-- ------------------------------------------------------ //-->
      <!-- columns for navigation accordion                       //-->
      <!-- ------------------------------------------------------ //-->
      <div class="col-xl-3 left">
        <!-- fruschtique logo //-->
        <div id="logo" class="bg-frBlue p-3"...>
        <!-- heading //-->
        <h4 class="mt-5 pointer">Graph <span class="font-italic"><span class="rcpCoII"></span></span></h4>
        <!-- accordion for navigation //-->
        <div id="accordion" role="tablist" class="mt-4">
          <!-- Quelle //-->
          <div class="card"...>
          <!-- Farblegende //-->
          <div class="card"...>
          <!-- Interaktion //-->
          <div class="card"...>
          <!-- Diagrams //-->
          <div class="card"...>
          <!-- Deutung //-->
          <div class="card"...>
          <!-- Community detection //-->
          <div class="card"...>
        </div>
        <!-- list of selected recipes //-->
        <div id="rcpByIgtList" class="list-group list-group-flush mt-4 mb-4"></div>
        <!-- link to index page //-->
        <a href="index.html" class="text-decoration-none"...>
      </div>
      <!-- ------------------------------------------------------ //-->
      <!-- columns for svg graphs                                 //-->
      <!-- ------------------------------------------------------ //-->
      <div id="svgDIV" class="col-xl-9 shadow bg-white" style="height: auto"...>
        <!-- ------------------------------------------------------ //-->
        <!-- out-of-flow spaces for diagrams and other //-->
        <!-- ------------------------------------------------------ //-->
        <!-- fact sheet //-->
        <div id="factSheet" class="chartArea col-xl-6 stretchRight tr shadow p-3 mb-5 bg-white rounded" style="visibility: hidden"...>
        <!-- diagram space for prevalence diagram //-->
        <div id="prevChartArea" class="chartArea col-xl-7 stretchRight tr shadow p-3 mb-5 bg-white rounded" style="visibility: hidden"...>
        <!-- diagram space for neighbor count diagram //-->
        <div id="neighChartArea" class="chartArea col-xl-7 stretchRight tr shadow p-3 mb-5 bg-white rounded" style="visibility: hidden"...>
        <!-- diagram space for degree diagram //-->
        <div id="degreeChartArea" class="chartArea col-xl-7 stretchRight tr shadow p-3 mb-5 bg-white rounded" style="visibility: hidden"...>
        <!-- diagram space for betweenness diagram //-->
        <div id="betweenChartArea" class="chartArea col-xl-7 stretchRight tr shadow p-3 mb-5 bg-white rounded" style="visibility: hidden"...>
        <!-- diagram space for scatter plot matrix //-->
        <div id="scatterChartArea" class="chartArea col-xl-7 stretchRight tr shadow p-3 mb-5 bg-white rounded" style="visibility: hidden"...>
        <!-- diagram space for multiple edges bar chart //-->
        <div id="multipleEdgesChartArea" class="chartArea col-xl-7 stretchRight tr shadow p-3 mb-5 bg-white rounded" style="visibility: hidden"...>
        <!-- diagram space for multiple edges bubble diagram //-->
        <div id="heatmapChartArea" class="chartArea col-xl-7 stretchRight tr shadow p-3 mb-5 bg-white rounded" style="visibility: hidden"...>
        <!-- diagram space for shared ingredients diagram //-->
        <div id="siChartArea" class="chartArea col-xl-7 stretchRight tr shadow p-3 mb-5 bg-white rounded" style="visibility: hidden"...>
        <!-- diagram space for Lorenz curve //-->
        <div id="LorenzCurveIngredientsArea" class="chartArea col-xl-7 stretchRight tr shadow p-3 mb-5 bg-white rounded"...>
        <!-- diagram space for clustering coefficients diagram //-->
        <div id="clCoChartArea" class="chartArea col-xl-7 stretchRight tr shadow p-3 mb-5 bg-white rounded" style="visibility: hidden"...>
      </div>
    </div>
    <datalist id="rcpNames"></datalist>
    <datalist id="igtNames"></datalist>
    <!-- diagrams //-->
    <script src="js/vendor/plotly-latest.min.js"></script>
    <!-- jsnx //-->
    <script src="js/vendor/jsnetworkx.js"></script>
    <!-- svg //-->
    <script src="js/vendor/snap.svg-min.js"></script>
    <!-- statistics //-->
    <script src="https://unpkg.com/simple-statistics@7.1.0/dist/simple-statistics.min.js"></script>
    <!-- My scripts //-->
    <script src="js/showMaster.js"></script>
    <!-- bootstrap and jQuery //-->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
            integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js"
            integrity="sha384-Piv4xVNRyMGpqkS2by6br4gNJ7DXjqk09RmUpJ8jgGtD7zP9yug3goQfGII0yAns" crossorigin="anonymous"></script>
  </body>
```

fruschtique
by Norbert Luttenberger


Thank you!