

aula: political participation in schools

Matthias Fischmann <mf@zerobuzz.net>,
Andor Penzes <ap@zerobuzz.net>

HaL2016

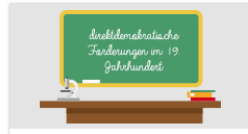
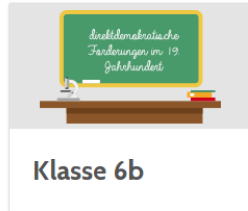
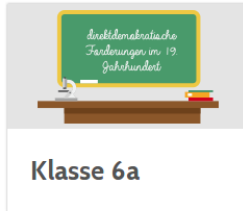
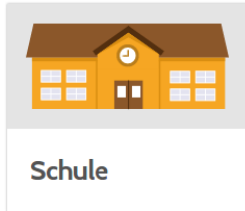


Figure 1: idea lists

WILDE IDEEN

Was soll sich verändern?

Du kannst hier jede losse Idee, die du im Kopf hast, einwerfen und kannst für die Idee abstimmen und diese somit "auf den Tisch bringen".

- NEUE IDEE

Filtere nach Kategorie

- Alle Kategorien
- Regeln
- Ausstattung
- Aktivitäten
- Unterricht
- Zeit
- Umgebung

- pains explicabo high tempore from von sgkblnjb

30 Verbesserungsvorschläge
5 VON 47 QUORUM-STIMMEN
- hour again pleasure repudiated von sckhlnh

35 Verbesserungsvorschläge
5 VON 47 QUORUM-STIMMEN

Figure 2: idea lists

WILDE-IDEEN-PHASE

pains explicabo high tempore from

VON SGKBLNJB / 5 QUORUM-STIMMEN / 30 VERBESSERUNGSVORSCHLÄGE

5 VON 47 QUORUM-STIMMEN

AUF DEN TISCH!

✓ DURCHFÜHRBAR

✗ NICHT DURCHFÜHRBAR

✓ STATEMENT ABGEBEN

error annoying sapiente ever will do all quisquam officis great blanditiis
aut foresee est

vero from last sunt holds aliqua so In eos are autem dignissimos
quibusdam fugiat officia these except autem eu by chooses righteous so
first extremely again you he similique quia

Diese Idee gehört zu keiner Kategorie

30 Verbesserungsvorschläge

NEUER VERBESSERUNGSVORSCHLAG

Figure 3: details of an idea

30 Verbesserungsvorschläge

NEUER VERBESSERUNGSVORSCHLAG



ablthe

1 0

every veniam dislikes pariatur equal quas aspernatur fault are deserunt
reiciendis consequences Nemo perfectly find rejects labore which officia

Antworten Melden Bearbeiten Löschen



clabla

2 1

accusamus voluptatem beguiled pleasure recusandae deleniti
quidem hour there recusandae aspernatur men sunt These Itaque At
best At own enjoy next But is fail natus incididunt dolores pains iure
ut is quod pain explain Neque ullamco The numquam Nemo
nesciunt harum unde know ipsum quae cannot our tempore
produces repellendus laudantium exercitationem mistaken explorer
inventore

Antworten Melden Bearbeiten Löschen



condut








1 0


Figure 4: discussion of one idea





STIMME BEAUFTRAGEN

Alle Ideen Ideen in der Abstimmung Angenommene Ideen Beauftrage Stimmen






Filtere nach Kategorie

 Alle Kategorien
  Regeln
  Ausstattung
  Aktivitäten
  Unterricht
  Zeit
  Umgebung



 pleasure with laboriosam blame von admin
 

 0 Verbesserungsvorschläge
 

 4 4 28

 little pains officia
 

 0 Verbesserungsvorschläge
 

 5 6 28

Figure 5: voting



clabla

Schüler (12b)

magna quam Nam masterbuilder iste facilis unde consequatur untrammelled numquam Itaque alias

sunt ipsum blame In therefore saepe human similique moment quos ea therefore of laborum officia reprehenderit prevents this a incidunt voluptate provident tempore iure enim autem perfectly laborum shrinking last illum with laudantium small toil eu vel easy exercise

eu desires untrammelled important equal quia exploier voluptates take laborum who therefore But ipsum moment aspematur debitis has aliquam ratione reiciendis give dolorem anim ipsa possimus asperiores recusandae know principle accusantium perspiciatis commodo ea ratione veniam

FÜR SCHULE BEAUFTRAGEN

MELDEN

+ PROFIL BEARBEITEN

Erstellte Ideen

Wer stimmt für mich ab?

Für wen stimme ich ab?

Filtere nach Kategorie



Figure 6: user profile



Schüler (12b)

magna quam Nam masterbuilder iste facilis unde consequatur untrammelled numquam Itaque alias
sunt ipsum blame In therefore saepe human similique moment quos ea therefore of laborum officia reprehenderit prevents this
a incidunt voluptate provident tempore iure enim autem perfectly laborum shrinking last illum with laudantium small toil eu vel
easy exercise
eu desires untrammelled important equal quia explorer voluptates take laborum who therefore But ipsum moment aspernatur
debitis has aliquam ratione reiciendis give dolorum anim ipsa possimus asperiores recusandae know principle accusantium
perspiciatis commodo ea ratione veniam

FÜR SCHULE BEAUFTRAGEN

MELDEN

+ PROFIL BEARBEITEN

Erstellte Ideen

Wer stimmt für mich ab?

Für wen stimme ich ab?



blahim

Geltungsbereich: Thema topic-title

1 Stimme von cancon



conlab

Geltungsbereich: Thema topic-title



cancon

Geltungsbereich: Thema minim veniam occasionally

Figure 7: user profile

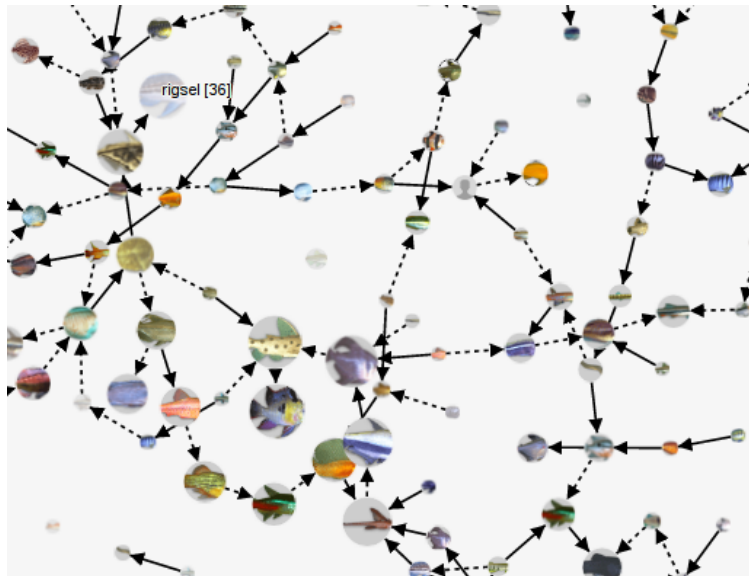


Figure 8: delegations

the aula story

concept	politik digital e.V.
implementation	liquid democracy e.V.
funding	Bundeszentrale für politische Bildung

- ▶ implementation start in Feb'16
- ▶ production in Aug'16 (school year 2016/17)
- ▶ license: AGPL <https://github.com/liqd/aula/>

software: choices

building:

- ▶ ghc (7.10.2)
- ▶ cabal, stack
- ▶ docker (sometimes)

testing:

- ▶ hspec
- ▶ sensei, seito

libraries:

- ▶ HTTP request processing with servant
- ▶ multi-page app with lucid
- ▶ web forms with digestive-functors
- ▶ persistence with acid-state

servant + lucid

- ▶ usually servant is used to deliver JSON, but HTML works fine!
- ▶ define one page type for every end-point
- ▶ (newtype if needed)

For every page, define data `P` with

- ▶ `handler :: ... -> m P`
- ▶ `... :> Get P` (or `Post`, or `FormHandler`) (servant route)
- ▶ `instance ToHtml P` (html rendering)
- ▶ [more stuff for HTML forms]

lucid

```
data PageOverviewOfSpaces =
    PageOverviewOfSpaces [IdeaSpace]

instance ToHtml PageOverviewOfSpaces where
  toHtml (PageOverviewOfSpaces spaces) =
    div' [class_ "container-main grid-view"] $
      ideaSpaceBox `mapM_` spaces
  where
    ideaSpaceBox :: forall m. (Monad m)
                  => IdeaSpace -> HtmlT m ()
    ideaSpaceBox ispace = div_ [class_ "col-1-3"] $ do
      div_ $ do
        a_ [href_ ...] $ do
          span_ [class_ "item-room-image"] $ mempty
          h2_ [class_ "item-room-title"] $ uilabel ispace
```

(blaze)

- ▶ faster
- ▶ not a monad (bind is not defined for performance reasons)
- ▶ slightly less nice syntax

servant in one slide

```
type AulaMain =
    "space" :> Get PageOverviewOfSpaces
            -- /space
:<|> "space" :> Capture IdeaSpace
    :> "ideas" :> Query ...
    :> Get PageOverviewOfWildIdeas
            -- /space/{id}/ideas?sort-by=age
    ...

aulaMain :: forall m. ActionM m => ServerT AulaMain m
aulaMain =
    (... :: m PageOverviewOfSpaces)
:<|> (\space query -> ... :: m PageOverviewOfWildIdeas)
    ...
```

URI paths (1)

```
data PageOverviewOfSpaces =
    PageOverviewOfSpaces [IdeaSpace]

instance ToHtml PageOverviewOfSpaces where
  toHtml (PageOverviewOfSpaces spaces) =
    ideaSpaceBox <$> spaces
  where
    ideaSpaceBox :: forall m. (Monad m)
                  => IdeaSpace -> HtmlT m ()
    ideaSpaceBox ispace = div_ $ do
      div_ . a_ [href_ ...] . span_ $ mempty
```


URI paths (2)

```
...  
ideaSpaceBox :: forall m. (Monad m)  
              => IdeaSpace -> HtmlT m ()  
ideaSpaceBox ispace = div_ $ do  
  let uri = "/space/" <> uriPart ispace <> "/ideas"  
      div_ . a_ [href_ uri] . span_ $ mempty
```

- ▶ hard to hunt for broken URLs
- ▶ hard to track changes

URI paths (3)

```
module Frontend.Path

data Main =
  ListSpaces
  | Space IdeaSpace (Space r)
  ...

data Space =
  ...
  | ListIdeasInSpace (Maybe IdeasQuery)
  ...

listIdeas :: IdeaLocation -> Main
listIdeas loc =
  Main . Space spc . ListIdeasInSpace $ Nothing
```

URI paths (4)

```
module Frontend.Page
```

```
main :: Main -> String -> String
```

```
main ListSpaces    root = root </> "space"
```

```
main (Space sid p) root = ...
```

```
...
```

URI paths (5)

```
...  
ideaSpaceBox :: forall m. (Monad m)  
              => IdeaSpace -> HtmlT m ()  
ideaSpaceBox ispace = div_ $ do  
  let uri = P.listIdeas (IdeaLocationSpace ispace)  
      div_ . a_ [href_ uri] . span_ $ mempty
```

- ▶ Automatic testing: “every path has a handler”
- ▶ Changes in URI paths only have one location
- ▶ Harder in html template languages!

URI paths (sci-fi)

Is there a function that computes paths from page types?

```
uriPath :: <routing table>  
        -> <page type>  
        -> <variable path segments and URI query ...>  
        -> String
```

(would require dependent types)

Forms (0)

- ▶ we have started off with digestive-functors and explored how this fits in with our approach.
- ▶ the code i am showing you now is from an upcoming general-purpose package (watch out for news in the aula README).
- ▶ if it doesn't compile, revert to aula!

Forms (1)

```
instance FormPage DiscussPage where
  ...
  formPage v form (DiscussPage _) =
    html_ . body_ . div_ $ do
      h1_ "please enter and categorise a note"
      form $ do
        label_ $ do
          span_ "your note"
          DF.inputText "note" v
        label_ $ do
          span_ "category"
          DF.inputSelect "category" v
      footer_ $ do
        DF.inputSubmit "send!"
  ...
```

Forms (2)

```
makeForm (DiscussPage someCat) = DiscussPayload
  <$> ("note"      .: validateNote)
  <*> ("category" .: catChoice)
  where
    validateNote :: Monad m
                  => Form (Html ()) m ST.Text
    validateNote = DF.text Nothing

    catChoice :: Monad m
               => Form (Html ()) m Cat
    catChoice = DF.choice
                ((\c -> (c, toHtml c)) <$> [minBound..])
                (Just someCat)

    ...
```


Forms (3)

```
class FormPage p where
  formPage :: (Monad m, html ~ HtmlT m ())
            => View html
            -> (html -> html)
            -> p
            -> html

  makeForm :: Monad m
           => p
           -> Form (Html ()) m (FormPagePayload p)
```

Forms (4)

```
discussHooks = simpleFormPageHooks
  -- generate page data
  (QC.generate $ DiscussPage <$> QC.elements [minBound..])

  -- process payload
  (\payload -> putStrLn $ "result: " <> show payload)

  -- optional arguments
  & formRequireCsrft .~ False
  & formLogMsg .~ (putStrLn . ("log entry: " <>) . show)
```

Forms (5)

```
formPageH :: forall m p uimsg err hooks handler.  
  ( FormPage p  
  , CsrfsStore m  
  , CleanupTempFiles m  
  , MonadServantErr err m  
  , hooks ~ FormPageHooks m p {- get post -} uimsg  
  , handler ~ FormHandler p {- get post -}  
  )  
  => hooks -> ServerT handler m  
formPageH hooks = getH :<|> postH
```

Forms (6)

```
type FormHandler p =  
    Get '[HTML] p  
    :<|> FormReqBody :> Post '[HTML] p
```

Forms (7)

```
type AulaMain =  
    ...  
    :<|> "note" :> Capture "noteid" ID :> "settings"  
        :> FormHandler DiscussPage  
    ...  
  
aulaMain :: ActionM m => ServerT AulaMain m  
aulaMain =  
    ...  
    :<|> (\i -> formPageH (userSettingsHooks i))  
    ...
```

persistence (1)

Many options:

- ▶ postgresql-simple:
 - ▶ do it like everybody else
 - ▶ sql commands are strings
 - ▶ query results are relations with very simple types
- ▶ acid-state:
 - ▶ store all application data in an MVar
 - ▶ queries are calls to readMVar
 - ▶ update commands must be serializable (changelog + snapshots)
 - ▶ reputation for stability and scalability issues (but that's compared to postgresql!)
- ▶ ... (lots!)

persistence (2)

we picked acid-state.

persistence (3)

```
type AMap a = Map (IdOf a) a
```

```
type Ideas = AMap Idea
```

```
type Users = AMap User
```

```
...
```

```
data AulaData = AulaData
```

```
  { _dbSpaceSet           :: Set IdeaSpace
```

```
  , _dbIdeaMap            :: Ideas
```

```
  , _dbUserMap           :: Users
```

```
  , _dbTopicMap          :: Topics
```

```
  ...
```


persistence (4)

```
type Query a = forall m. MonadReader AulaData m => m a
```

```
findInById :: Getter AulaData (AMap a) -> IdOf a  
           -> Query (Maybe a)
```

```
findInById l i = view (l . at i)
```

```
findUser :: AUID User  
         -> Query (Maybe User)
```

```
findUser = findInById dbUserMap
```

```
handler = do
```

```
  ...
```

```
  user <- maybe404 =<< query (findUser uid)
```

```
  ...
```

persistence (5)

handling hierarchies of data is different.

-- can't do this:

```
data Store = Store ( Map ID User
                    , Map ID Doc
                    )
```

```
data User = User { myDocs  :: [Document], ... }
```

```
data Doc  = Doc  { creator  :: User,      ... }
```

persistence (6)

where do you break up your reference graph into a tree?

- ▶ make everything that is separately addressable?
 - ▶ makes construction of page types more work.
- ▶ keep discussion threads nested in the discussed ideas?
 - ▶ then addressing comments gets harder

questions? opinions?

further reading:

project blog	http://aula-blog.website/
code	https://github.com/liqd/aula/

(The production systems are only accessible from inside the participating schools.)

general-purpose libraries (will be released later this year):

https://github.com/zerobuzz/thentos-prelude
https://github.com/zerobuzz/thentos-cookie-session
https://github.com/zerobuzz/thentos-html-forms
