

building web applications with lucid, servant, digestive-functors

Matthias Fischmann <mf@zerobuzz.net>,
Andor Penzes <ap@zerobuzz.net>,
Nicolas Pouillard np@nicolaspouillard.fr>

Haskell Exchange 2016

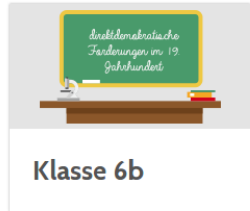
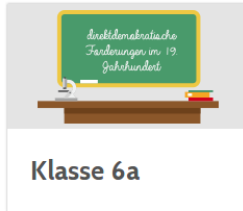
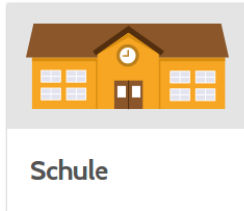


Figure 1: idea spaces

aula Start Beauftragungen Hi admin

Wilde Ideen der Schule Themen auf dem Tisch der Schule

WILDE IDEEN

Was soll sich verändern?

Du kannst hier jede lose 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 sckhlnih

35 Verbesserungsvorschläge
5 VON 47 QUORUM-STIMMEN

Figure 2: idea lists

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 officiis 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

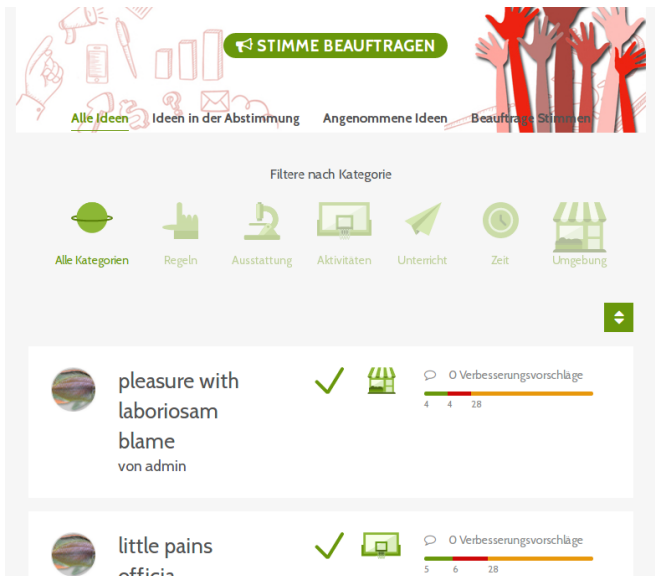


Figure 5: voting



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 inure enim autem perfectly laborum shrinking last illum with laudantium small toil eu vel
easy exercise
eu desires untrammelled important equal quia explore 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

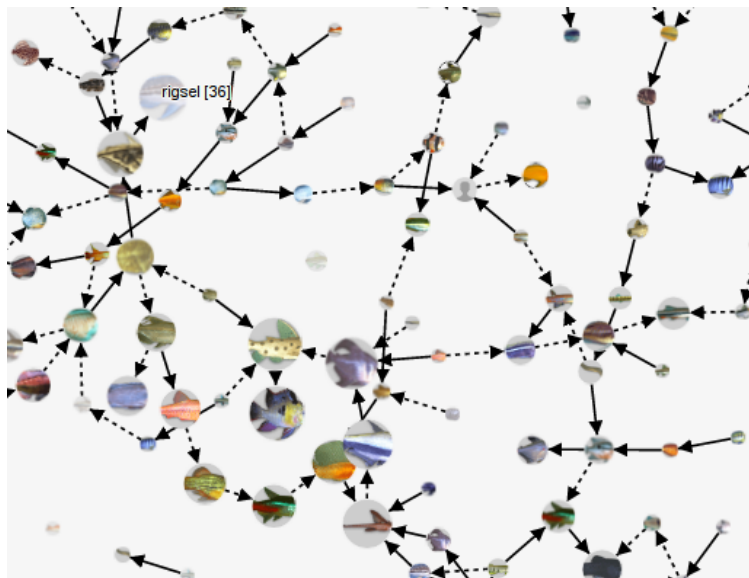


Figure 8: delegations

- ▶ <https://github.com/liqd/aula/>
- ▶ license: AGPL

software: choices

libraries:

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

testing:

- ▶ hspect
- ▶ sensei, seito

building:

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

lucid

```
data PageOverviewOfSpaces =  
    PageOverviewOfSpaces [IdeaSpace]
```

lucid

```
data PageOverviewOfSpaces =  
    PageOverviewOfSpaces [IdeaSpace]  
  
instance ToHtml PageOverviewOfSpaces where  
    toHtml (PageOverviewOfSpaces spaces) =  
        div' [class_ "container-main grid-view"] $  
            a_ [href_ ...] $  
                "click me"
```

lucid

```
data PageOverviewOfSpaces =
    PageOverviewOfSpaces [IdeaSpace]

instance ToHtml PageOverviewOfSpaces where
  toHtml (PageOverviewOfSpaces spaces) =
    div' [class_ "container-main grid-view"] $
      ideaSpaceBox `mapM_` spaces

ideaSpaceBox :: forall m. (Monad m)
              => IdeaSpace -> HtmlLT 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

```
"space" :> Get PageOverviewOfSpaces  
        -- /space
```


servant in one slide

```
"space" :> Capture IdeaSpace  
  :> "ideas" :> Query ...  
    :> Get PageWildIdeas  
      -- /space/7a/ideas?sort-by=age
```

servant in one slide

```
type AulaMain =  
    "space" :> Get PageOverviewOfSpaces  
            -- /space  
:<|> "space" :> Capture IdeaSpace  
    :> "ideas" :> Query ...  
    :> Get PageWildIdeas  
            -- /space/7a/ideas?sort-by=age  
...
```

servant in one slide

```
type AulaMain =
    "space" :> Get PageOverviewOfSpaces
              -- /space
:<|> "space" :> Capture IdeaSpace
      :> "ideas" :> Query ...
      :> Get PageWildIdeas
              -- /space/7a/ideas?sort-by=age
...

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

servant + lucid

- ▶ usually servant is used to deliver JSON, but HTML works fine!

for every page

- ▶ page type
- ▶ servant end-point
- ▶ ToHtml instance

in addition, for forms

- ▶ special type alias for pair of GET, POST end-points
- ▶ FormPage instance (more on that in a moment)

Forms (0)

we need to

- ▶ describe the shape of the form contents
- ▶ render (*view*) the form
- ▶ parse (*validate*) user input

we want

- ▶ independent parts
- ▶ composability

Forms (1)

```
data DiscussPage = DiscussPage Cat
```

```
data Cat = Yay | Nay | Noise
```

Forms (2)

```
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 (3)

```
makeForm (DiscussPage cat) = 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 cat)
```


Forms (4)

```
class FormPage p where
  formPage :: View (Html ())
             -> (Html () -> Html ())
             -> p
             -> Html ()

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

Forms (5)

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

Forms (6)

```
type FormHandler p =  
    Get '[HTML] p  
    :<|> FormReqBody :> Post '[HTML] p  
  
formPageH :: (FormPage p, ...)  
    => FormPageHooks m p  
    -> ServerT (FormHandler p) m  
formPageH hooks = (... :<|> ...)
```

Forms (7)

```
type AulaMain =  
  ...  
  :<|> "note" :> Capture "noteid" ID  
        :> FormHandler DiscussPage  
  ...  
  
aulaMain :: ActionM m => ServerT AulaMain m  
aulaMain =  
  ...  
  :<|> (formPageH . discussHooks)  
  ...
```

Forms (8)

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

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

Forms (9)

```
discussHooks _i = 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)
```

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 (servant style)

```
type AulaMain =  
    ...  
    "space" :> Get PageOverviewOfSpaces  
            -- /space  
    ...  
  
overviewOfPagesPath :: URI  
overviewOfPagesPath = safeLink  
    (Proxy :: Proxy AulaMain)  
    (Proxy :: Proxy ("space" :> Get PageOverviewOfSpaces))
```

URI paths (servant style)

```
type AulaMain =  
  ...  
    "space"  :> Capture IdeaSpace  
      :> "ideas" :> Query ...  
        :> Get PageWildIdeas  
          -- /space/7a/ideas?sort-by=age  
  ...  
  
wildIdeasPath :: IdeaSpace -> ... -> URI  
wildIdeasPath = safeLink  
  (Proxy :: Proxy AulaMain)  
  (Proxy :: Proxy ("space" :> Capture IdeaSpace :> ...))
```

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)

thanks you!

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

<https://github.com/zerobuzz/thentos-cookie-session>

<https://github.com/zerobuzz/thentos-html-forms>

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.)