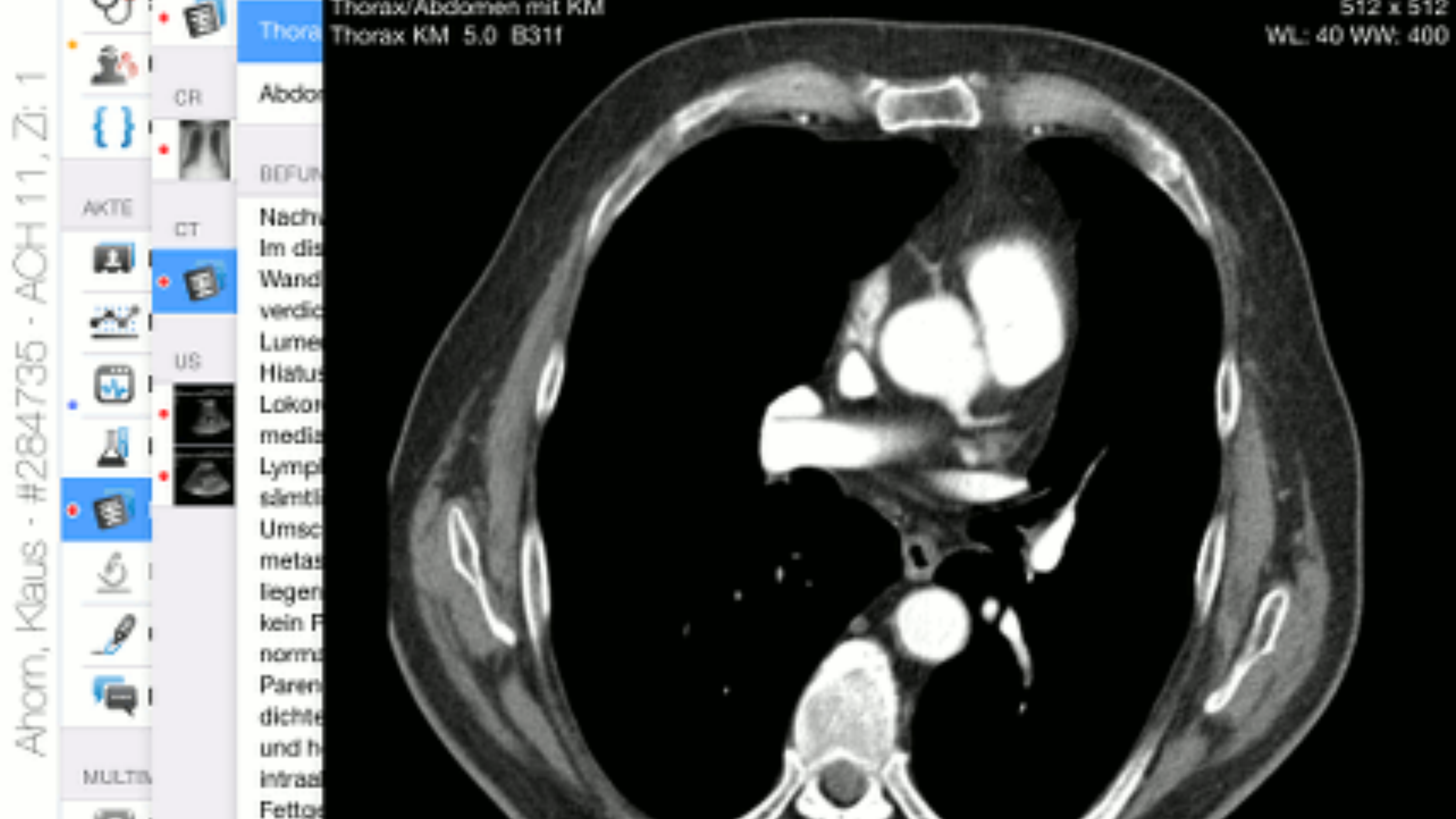
SPOCK

POWERFUL ELEGANT WEB APPLICATIONS USING *Haskell*

ALEXANDER THIEMANN

- Checkpad MED
- TramCloud
- Bahn-Buddy.de
- Computer Science Student



Nach
Im dis
Wand
verdic
Lume
Hiatus
Lokor
medis
Lympl
sämtli
Umsc
metas
liegen
kein F
norma
Paren
dichte
und h
intraa
Fettge

```php
<?php
function isUserLoggedIn($user, $password) {
    $res = mysql_query("SELECT * FROM user WHERE username = ".$user);
    if ($res) {
        echo "Oh, hello $user";
    }
    // ...
}
?>
```

PHP MISTAKES ?
MEMECAPTAIN

HASKELL

1. quick development
2. fast iteration
3. high performance
4. good scalability
5. reliable
6. robust
7. secure

SPOCK?

"BENEFIT AS MUCH AS POSSIBLE FROM TYPES WITHOUT (MUCH) *type level programming*"

```haskell
main :: IO ()
main =
    runSpock 3000 $ spock cfg $
    do get ("add" <//> var <//> var) compute

compute :: Int -> Int -> ActionT m ()
compute a b = html $ renderHtml $ span_ (a + b)
```

```haskell
import Data.HVect

curry :: HasRep ts => (HVect ts -> a) -> HVectElim ts a
uncurry :: HVectElim ts a -> HVect ts -> a

-- Int -> Int -> ActionT m ()
--       <==> HVectElim '[Int, Int] (ActionT m ())
```

```haskell
addPath :: Path '[Int, Int]
addPath = "add" <//> var <//> var

get :: Path xs -> HVectElim xs (ActionT m ()) -> SpockT m ()
```

(LOGIC) BUGS: POTENTIAL SECURITY HOLES

27% (481) BUGS: USER INPUT SANITIZATION (HACKERONE)

EXAMPLE: 1.4% (24) BROKEN/OPEN AUTHENTICATION

CAN BE PREVENTED BY *types*!

```haskell
authHook :: ActionCtxT (HVect xs) m (HVect (User ': xs))
authHook =
    do oldCtx <- getContext
       sess <- readSession
       user <- getUser
       return (user :&: oldCtx)

getSecretData :: ListContains n User xs => ActionCtxT (HVect xs) m Secret
getSecretData = undefined -- ...

app =
    prehook authHook $
    do get "some-action" getSecretData
```

XSS 21.87% (375) CAN BE PREVENTED BY

*types**

SINGLE PAGE APPLICATIONS

everywhere

APIS

SEPARATION OF LOGIC AND VIEW

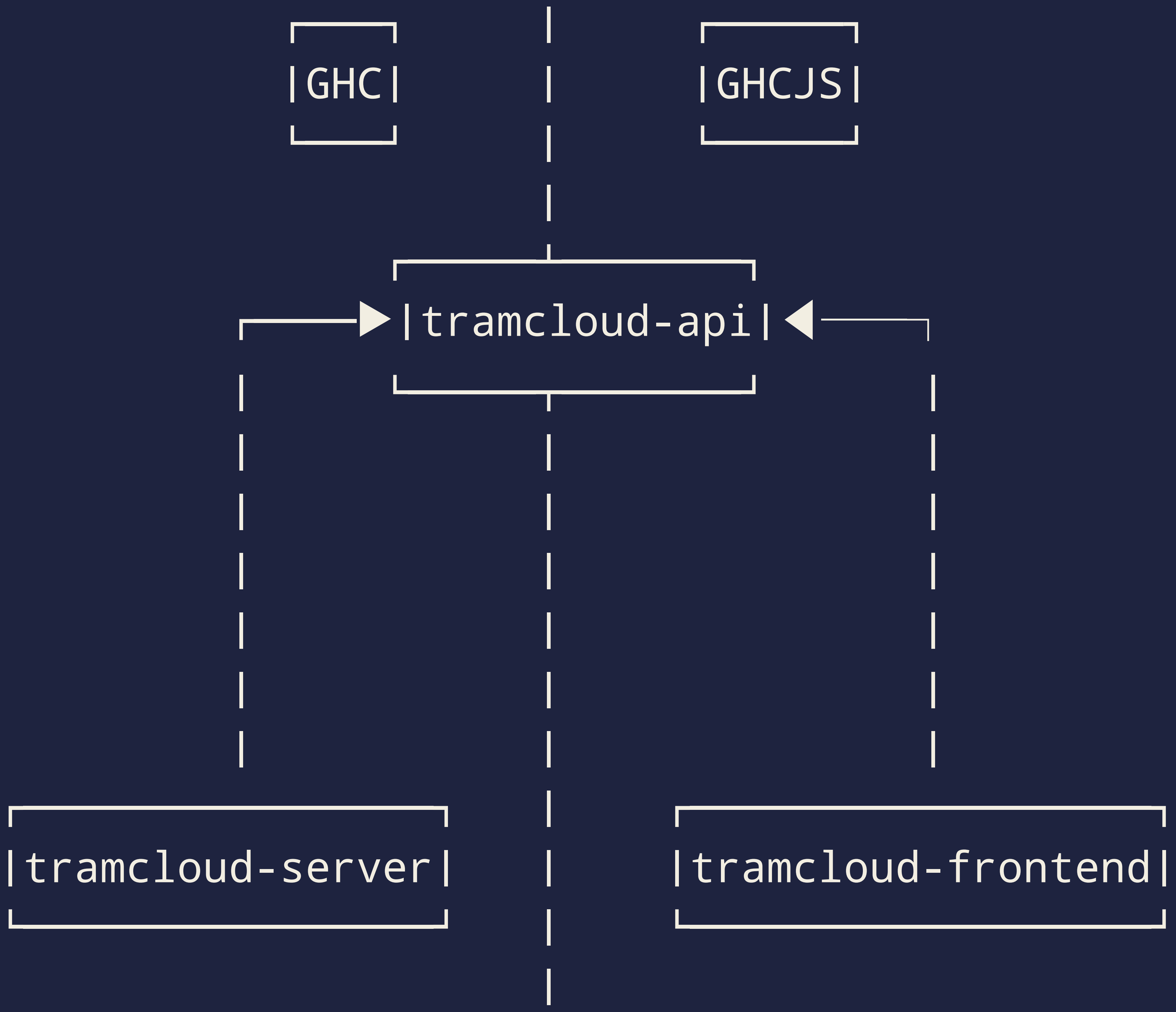
```haskell
-- remember?
compute :: Int -> Int -> ActionT m ()
compute a b = html $ renderHtml $ span_ (a + b)
```

```haskell
add :: Int -> Int -> Int
add = (+)

htmlRenderer = html . renderHtml . span_
jsonRenderer = \x -> json ["result" .= x]
```

HOW DO WE COMMUNICATE APIS?

GHCJS


GHC
GHCJS
tramcloud-api
tramcloud-server
tramcloud-frontend

```haskell
module ApiDef where

data LoginReq = LoginReq { username :: !T.Text , password :: !T.Text }
    deriving ({- ... -}ToJSON, FromJSON)

data LoginResp = LoginOkay | LoginFailed
   deriving ({- ... -}ToJSON, FromJSON)

loginUser :: Endpoint '[] ('Just LoginReq) LoginResp
loginUser = MethodPost Proxy ("api" <//> "user" <//> "auth")
```

```haskell
import ApiDef

api :: Application ()
api =
    defEndpoint loginUser loginHandler

loginHandler :: LoginReq -> Action LoginResp
loginHandler r =
    do auth <- runDB $ \conn -> authUser conn (username r) (password r)
       -- ...
       pure LoginFailed
```

```haskell
import ApiDef

do res <- callEndpoint loginUser (LoginReq user pass)
   -- ...
```

DEPLOYMENT / COMPATIBILITY

- JSON: only add optional fields
- use protocol buffers
- version your APIs

USE TYPES FROM *-API PACKAGE INTERNALLY?

BUILDING

- two stack files stack.yaml and stack-ghcjs.yaml
- (optional) "link" GHCJS output with browserify
- GHGJS output with closure-compiler or uglifyJS

THERE'S MORE...

- fast typesafe routing
- middleware
- sessions
- cookies
- database helper
- csrf-protection
- typesafe contexts

CHOICE OF LIBRARY / FRAMEWORK?

QUESTIONS?