Ein Gopherserver in Haskell

Viel Liebe!

sternenseemann
<lambda@lukasepple.de>

5. November 2015

Gopher?

- Gopher: 1993
- Hierarchische Struktur (Read-Only Filesystem!)
- 15 Seiten RFC
- Untot.

- HTTP/1.0: 1996
- Hyperlinks und lustiges Herumspringen
- 60 Seiten RFC
- Wichtig!

Das Protokoll

```
Client: {Öffnet Verbindung}

Server: {Akzeptiert die Verbindung}

Client: <Selektor>\r\n

Server: {Gibt die selektierte Datei oder
         ein Gopher Menu zurück          }
```

Selektor: Unix-Pfad oder ein leerer Selektor, der / entspricht.

Gopher Menus

Format:

```
{Dateityp}{Titel}\t{Pfad}\t{Hostname}\t{Port}\r\n
```

Gopher Menus

Format:

```
{Dateityp}{Titel}\t{Pfad}\t{Hostname}\t{Port}\r\n
```

Beispiel:

```
gLustige Katze  /kadse.gif  localhorst.org  70
```

Gopher Menus

Format:

```
{Dateityp}{Titel}\t{Pfad}\t{Hostname}\t{Port}\r\n
```

Beispiel:

```
gLustige Katze  /kadse.gif  localhorst.org  70
```

- 1: Directory
- 2: Phone Book Server
- 3: Error
- 4: Binhex Macintosh File
- 5: DOS Archive
- 6: Unix uuencoded File
- 7: *Index Search Server*
- 8: Telnet Session
- 9: Binary File
- +: *Redundant Server*
- T: Tn3270 Session
- g: *Gif File*
- I: Image File

Protokollerweiterungen

- Der i-Typ: Nur-Text-Einträge im Menü.
- Gophermap: Dateiformat, um Menüs zu beschreiben.
- Gopher+: Verbesserungen aus dem Jahr von RFC1436.

Spacecookie

- Dateiservierer über Gopher
- Privilegien werden abgegeben (mit `System.Posix.User`).
- 100% systemd-kompatibel!
- Gophermaps und i-Typ
- Haskell!
- Kein HTTP-Server integriert.

Intern

```
data GopherFileType = File
                    | ...

data GopherMenuItem = Item GopherFileType ByteString
  GopherPath ByteString PortNumber

data GopherResponse = MenuResponse [GopherMenuItem]
  | FileResponse ByteString
  | ErrorResponse ByteString ByteString PortNumber

data GophermapEntry = GophermapEntry GopherFileType
  ByteString (Maybe GopherPath)
  (Maybe ByteString) (Maybe PortNumber)
  deriving (Show, Eq)
```

Intern

```haskell
newtype Spacecookie a = Spacecookie
  { runSpacecookie :: ReaderT GopherdEnv IO a }
  deriving ( Functor, Applicative, Monad
           , MonadIO, MonadReader GopherdEnv)

data GopherdEnv = GopherdEnv { serverSocket :: Socket
                             , serverConfig :: Config
                             }

data Config = Config { serverName    :: ByteString
                     , serverPort    :: PortNumber
                     , runUserName   :: ByteString
                     , rootDirectory :: FilePath
                     }
```

Intern

```
requestToResponse :: GopherPath -> GopherFileType ->
  Bool -> Spacecookie GopherResponse

mainLoop :: Spacecookie ()
mainLoop = do
  env <- ask
  let sock = serverSocket env
  forever $ do
    (clientSock, _) <- liftIO $ accept sock
    liftIO $ forkIO $ (runReaderT . runSpacecookie)
     (handleIncoming clientSock) env
  liftIO $ cleanup sock
```

Demo!

Danke!

Konstruktiv auslachen:
`https://github.com/lukasepple/spacecookie`

Noch Fragen?

♥