Sympinátor

maturitní projekt

jméno: Ondřej Tkaczyszyn
třída: 4.A

Co to je?

Hledíte právě tváří v chřtán ruinám možná nejambicióznějšího maturitního projektu[1], který se kdy marně snažil najít si cestu z prstů Ondřeje Tkaczyszyna.

Má jít o projekt, který pomáhá s organizací přednáškového fóra Symposion, při kterém je potřeba dávat dohromady seznamy hostů a přednášek, vynášet nad nimi rozsudky, zapisovat data, která o nich postupně získáváme, a snažit se, aby se co nejdříve objevila na oficiálním webu akce, což typicky zatěžuje tvůrce webu akce pro daný rok. Právě na něj autor myslel, když se mu rodila v hlavě idea této aplikace -- právě tyto funkce (*navrhování, vyhodnocování, umist'ování do tabulky, organizování, interní komunikace)* má aplikace zastávat. Zdali v tomto úmyslu za dobu určenou ke svému zrodu stihla dosíci všech vytyčených met, nechť posoudí laskavý čtenář.

Ovšem další autorovou ambicí bylo vyzkoumat, jak se tvoří serverové programy, a to s co možná nejmenší závislostí na již dříve vytvořených různých knihovnách apod. -- část, která obstarává kontakt mezi serverem a uživatelem stojí s výjimkou bcryptu (který by bylo jistě nevhodné snažit se nahradit vlastním výtvorem) z většiny na základních knihovnách dostupných v Go.
Frontend pak slouží dvojí roli coby neohrabaný přístup k ovládání aplikace a umělecké dílo pozdně avantgardního rázu.

[1] To mu však nikterak neubírá na povšechné fádnosti.

Jak to nainstalovat?

prerekvizity:

- make (nebo schopnost číst Makefile a kopírovat do příkazové řádky)
- docker, docker-compose
- *npm (pro vývoj), pak ideálně i go*
- ideálně unixový operační systém, na Windows netestováno

spustit v pořadí příkazy:

git clone --recursive https://github.com/ondrax/sympinator-be
git submodule update --init --recursive
make up
make up & (cd frontend && npm run start) # pokud je třeba živě pracovat na frontendu

Jak to používat a co to umí?

Ačkoli se aplikace může zpočátku zdát vhodná spíše coby terč výsměchu, umí přeci jen několik věcí:

- vytvářet nové uživatele
  localhost:8000/app/ -> Založit účet
- kontrolovat správnost zadání příštupových údajů k stávajícím uživatelům
  localhost:8000/app/ -> přihlásit se jako jeden z:
  - user -- role *běžný uživatel*
  - root -- role *administrátor*
  - coordinator -- role *koordinátor*
  - editor -- role *editor*
- trochu nešikovným způsobem si lokálně (zatím bez možnosti ukládání do databáze [byl jsem blízko]) hrát se zápisy, které již jsou v systému uložené, pro
  - časový rozvrh akce
  - jednotlivé přednášky
  - navrhování přednášek (login->přednášky->navrhnout)
- nechá se to spustit v dockeru (frontend a backend zvlášť -- nevím už, jak pracují ty dva dohromady, normálně ale backend servuje frontend)
- VEŘEJNĚ! (samozřejmě většina těch endpointů má být skrytá za autentifikaci, jenom jsem to tak daleko zatím nedopracoval) -- vypsány v $BACKEND/code/api/main.go
  - většina vrací validní JSON, některé stringy logovací
- má to navržené schéma databáze (v $BACKEND/docs)

Ke zohlednění:

Bezpečnost:

+ heslo se hashuje a takové se ukládá v databázi (bcrypt2, salt)
- ježto jsem neplatil za certifikát, stejně se posílá nešifrované
- oprávnění (a to hodně ezoterickým způsobem) řeší jenom frontend, na backendu autentifikace hapruje, protože jsem se nenaučil používat middleware na routování
+ technicky ale frontend v reakci na login nějaký token dostává a technicky existuje spousta řádků kódu, které se o to starají (v $BACEKND/code/api/auth.go JWTAuthMiddleware -- v kombinaci s tím, že z frontendu se měly API requesty posílat výhradně pomocí $FRONTEND/app/SessionManagement.js, který udržuje informace o uživateli napříč Reactími změnami (ovšem kvůli nějaké chybě se stejně data mažou při znovunačtení stránky -- takhle localStorage fungovat nemá)
+ to málo Regexů, které aplikace používá, je chráněno proti ReDoSu (byť celá věc jistojistě mře na spoustu jiných slabin) hned dvojmo:
    + A) nejsou špatně napsány
    + B) go garantuje, že půjdou spustit v O(n) s n délkou vstupu