**Úvodné info:**

Backend je písaní v jazyku Golang(go). Backend potrebuje postgresql(s rozšírením plpgsql), v ktorom sú vytvorené tabuľky a funkcie podľa skriptov na githube.

Jazyk Golang sa dá stiahnuť tu: [The Go Programming Language (golang.org)](https://golang.org/).

V čase písania bola aktuálna verzia go 1.15.6, pokiaľ sa nič nezmení, tak všetky ďalšie verzie by mali byť kompatibilné – program sa môže skompilovať v novších verziách. Program bol vytvorený na platforme Windows(Amd64), ale je odskúšaný(bez úprav) aj na platforme Linux(Amd64-Ubuntu aj Arm64-Debian) a finálna verzia bola kompilovaná v go 1.17(každá verzia prináša zmenšenie binárky), mal by byť prenositeľný aj na platformu Mac, BSD, Solaris........, ale tie neboli v tomto projekte odskúšané.

Po stiahnutí a nainštalovaní sú hlavnými príkazmi(pre projekt):

1. go build

* tento príkaz skompiluje celý program, v comandline sa musí nachádzať v právnom priečinku „tisko“, ktoré sa musí nachádzať v GOPATH/src(odporúčam editor Goland)
* pre menšiu veľkosť binárneho súboru je možné použiť flag: ‚-ldflags „-w -s“(go build -ldflags „-w -s“, úspora +-25%), v extrémnej situácii aj samostatný program „upx“ s flagom –ultra-brute(úspora s go build flagmi +-75%) – daň je menšia rýchlosť pri štartovaní

1. go get

* tento príkaz slúži na sťahovanie knižníc 3. strán a všetkých závislostí, ktoré obsahujú
* Musí byť nainštalovaný git

1. go mod

* pre kompletnú inštaláciu závislostí sa použije vendor(go mod vendor)

Na gite sa nachádza kniznice.zip, v ktorom sú zazipované knižnice potrebné na skompilovanie, manuálne je možné ich vsunúť do projektu, či už ako GOPATH(cesta, kde sa nachádzajú projekty Golang) alebo ako „vendor“(lokálne projektové knižnice) , ale odporúčaný je vyššie uvedený go mod vendor.

**Poznámky, čo môžete využiť a na čo si dávať pozor:**

* Golang má ľahkú inter-kompatibilitu so starým C(volá sa to cgo a volá sa s príkazom import „C“ a nad tým C-eckovský(je potrebný gcc kompilátor) program-WARNING: v C-écku je Golangový garbage colektor vypnutý a samotné volania sú 100x pomalšie ako volanie klasickej funkcie), taktiež v projekte je už vytvorený spôsob ako sa s pomocou cmd môže volať ľubovoľný spustiteľný program, ale má to obmedzenia na počet vstupov a výstupov.
* V projekte sa niekedy objavuje slovko „go“, jazyk má aj „goto“ – nepomýliť! „go“ znamená vytvorenie gorutiny, ktorá sa trošku nepresne môže nazvať „odľahčený thread“ a „goto“ je prevzaté z C-éčka, aj keď má väčšie obmedzenia.
* V projekte sa vyskytujú aj „chaneli“, opatrne, môžu zablokovať pri nesprávnom použitý celé vlákno.
* V projekte sú dva typy skriptov 1. typ so znakom „?“, kde záleží na poradí argumentov a 2. „novší“ typ mená, ktoré sa v programe s príkazom „strings.ReplaceAll(string, old, new)“ nahradzujú.
* V ďalších častiach sa bude rozoberať backend, ale binárny súbor má za úlohu aj spustiť a poskytovať prístup k frontendu, ktorý je písaný v javascripte a v reacte, následne bol „zbuildovaný“ príkazom npm run-script build a daný do priečinka „build\_front\_end“ .
* Odporúčam naštudovať si zabudované struktúry - triedy – embeded struct – nepomýliť s package embed!

**Štruktúra:**

Backend je(okrem hlavného súboru main.go, ktorý spúšťa program a niektorých súborov pozrite sa najmä na gefco.log, kde sú výpisy chýb) rozdelený do package-ov, niektoré importujú iné(pozor na zakázané cyklické importovanie, ktoré nie je dovolené). V Golangu sú štruktúry a funkcie s VEĽKÝMI písmenami verejné(viditeľné mimo package) a s malými privátne. Každý package má svoje SQL príkazy uložené v .sql(.txt, .json alebo .lock) súboroch, väčšina z nich je v pod-package „scripts“, v tomto adresári môžu byť aj konfiguračné súbory, ktoré sa po spustení programu načítajú do pamäte bol to pokus urobiť SQL a kód nezávislými, ale kód potrebuje špecifické skripty, preto odporúčam ich brať ako celok, pretože niektoré skripty sa používajú vo viacerých funkciách ( napríklad: filter\_training.sql sa používa na vyhľadávanie podľa filtra, ale rovnaký skript s úpravou za behu sa používa na všetky trainingy [Query1 Query2 sa nahradia prázdnymi stringami]), tu je ich zoznam a popis:

1. **helper**

* Komponenty(funkcie a štruktúry-triedy), ktoré sa používajú v mnohých iných(skoro všetky, preto nebude spomínaný)
* Neobsahuje SQL skripty
* V zdrojovom súbore „run\_scripts.go“ sa nachádza rozhranie pre komunikáciu s akýmkoľvek spustiteľným súborom
* Tu sa nachádza aj komponent „fork“, ktorý spúšťa pomocou command line sám seba, keď „dieťa“ skončí, tak sa zapíše chyba a spustí sa znovu

1. **paths**

* Všetky mená subdomien, ktoré sa v programe vyskytujú, POZOR: bez metód a koncových znakov
* Obsahuje aj „GlobalDir“, ktorým sa nastavuje cesta z ktorej binárna vychádza
* Neobsahuje SQL skripty

1. **tiker**

* Primitívny časovač
* Neobsahuje SQL skript

1. **combination**

* Poskytovanie prístupu k tabuľkám „branch“, „city“, „department“,  „division“ a ich 4-kombináciam
* Skript na získanie platných kombinácií(*combinations.txt*)

1. **connection\_database**

* Databázové spojenie k databáze
* Skripty na stránku, ktorá obsahuje všetky rozhrania, ktoré sú poskytované front endu(*begin\_homepage.html,end\_homepage.html*)
* Servery http obsluhujúci všetky domény(tu sa registrujú pri začatí všetky domény a následne sa spustí server plniaci dve úlohy:

1. Backend so všetkými jeho doménami – cieľ tohto dokumentu
2. (Frontend so všetkými stránkami – vedľajšie)

* Súbor obsahujúci port frontendu a backendu(*port.txt*)
* Konfig do databázy(*postgres\_config.txt*)
* (Nebudem spomínať jeho potrebu, pretože skoro každý ho potrebuje)

1. **document**

* Vlastná štruktúra(trieda)
* CRU(bez D) operácie
* Pomocné skripty k CRU operáciám
* Zobrazovanie podľa kategórií aktual, edited, filter)
* Výnimka: podpisy sa riešia po potvrdení dokumentu tu !

1. **employee**

* Vlastná štruktúra(trieda)
* Prihlasovanie
* Poskytovanie pohľadov do databázy všetkých zamestnancov a filtrovaných podľa filtra
* Skripty ku pohľadom
* Konfig na dovolenie prihlasovania bez hesla

1. **~~exports~~**

* Nevyužitý package pre export
* Interakcia Pythonu s Golangom(výsledok Pythonu sa ukladal sem)

1. **languages**

* Nevyužitý(stále zabudovaný) package pre poskytovanie jazyka pre front end

1. **imports**

* Pomocný adresár pre importy
* Priečinok na interakciu Pythonu s Golangom(výsledok Golangu sa ukladal sem)

1. **mail**

* Odosielanie mailov
* Potrebuje package employee a tiker
* Kontrola a posielanie upozornení
* Skripty pre správy mailov
* Skripty konfigu(*mail.lock)*, ktorý slúži pri výpadkoch, aby program posielal raz do dňa *emails\_of\_admins.txt*, ktorý má mať maily adminov, kde má chodiť notifikácia *mail\_config.txt*, ktorý nastavuje server heslo meno a správy)

1. **signature**

* Podpisy – náhľad – podpísané, nepodpísané podpisy od online tréningov a dokumentov, skillmatrixy
* Podpisovanie, znovu podpísanie a zrušenie podpísania
* Potrebuje employee, document, training
* Modifikované štruktúry – premena z „fake“
* Posielanie podpisov novým zamesnancom
* pod-package:

1: fake\_struct – Štruktúry upravené na vyberanie z db

* Výnimka: confirm training sa robí tu!

1. **training**

* Vlastná štruktúra(trieda)
* Create, update
* Pohľad edited

1. **upload\_export\_files**

* ~~Export využíva Python a dáva výsledok k dispozícií~~
* Import parsuje zamestnancov a kartičky z csv(kódovanie utf8[default pre golang) do db (pozn. iné ako csv odmieta)
* Využíva package signuture
* Konfiguračný skript(*csv\_config.txt*) – nastavuje ktoré stĺpce sa majú považovať za príslušné údaje a ešte, ktoré keď sa nenájdu, tak sa má riadok ignorovať

**Rozhranie:** 1., Domovská stránka backendu poskytuje html stránku na obrázku, domény sú usporiadané podľa abecedy, domény majú vypísanú url, s ktorou sa zavolala s koncovkou a metódou, ktorou sa majú volať. 1., Domovská stránka backendu poskytuje html stránku na obrázku, domény sú usporiadané podľa abecedy, domény majú vypísanú url, s ktorou sa zavolala s koncovkou a metódou,

* **1 link: http://localhost:8180/auth/kiosk method: POST**
* **2 link: http://localhost:8180/auth/login method: POST**
* **3 link: http://localhost:8180/branches method: GET**
* **4 link: http://localhost:8180/cancels method: POST**
* **5 link: http://localhost:8180/cities method: GET**
* **6 link: http://localhost:8180/combinations method: GET**
* **7 link: http://localhost:8180/control7777777 method: POST**
* **8 link: http://localhost:8180/departments method: GET**
* **9 link: http://localhost:8180/divisions method: GET**
* **10 link: http://localhost:8180/document/actual method: GET**
* **11 link: http://localhost:8180/document/confirm/{id} method: GET**
* **12 link: http://localhost:8180/document/create method: POST**
* **13 link: http://localhost:8180/document/create/confirm method: POST**
* **14 link: http://localhost:8180/document/edited method: GET**
* **15 link: http://localhost:8180/document/filter method: POST**
* **16 link: http://localhost:8180/document/report/{id} method: GET**
* **17 link: http://localhost:8180/document/update method: POST**
* **18 link: http://localhost:8180/document/update/confirm method: POST**
* **19 link: http://localhost:8180/employees/all method: GET**
* **20 link: http://localhost:8180/employees/{filter} method: GET**
* **21 link: http://localhost:8180/file/export/{format}/{id} method: GET**
* **22 link: http://localhost:8180/file/upload method: POST**
* **23 link: http://localhost:8180/language/{language} method: GET**
* **24 link: http://localhost:8180/languages/all method: GET**
* **25 link: http://localhost:8180/lectors/all method: GET**
* **26 link: http://localhost:8180/resigns method: POST**
* **27 link: http://localhost:8180/sign/document method: POST**
* **28 link: http://localhost:8180/sign/superior method: POST**
* **29 link: http://localhost:8180/sign/training method: POST**
* **30 link: http://localhost:8180/signed/signatures/{id} method: GET**
* **31 link: http://localhost:8180/skill/matrix method: POST**
* **32 link: http://localhost:8180/training/all method: GET**
* **33 link: http://localhost:8180/training/confirm/{id} method: GET**
* **34 link: http://localhost:8180/training/create method: POST**
* **35 link: http://localhost:8180/training/create/confirm method: POST**
* **36 link: http://localhost:8180/training/edited method: GET**
* **37 link: http://localhost:8180/training/filter method: POST**
* **38 link: http://localhost:8180/training/report/{id} method: GET**
* **39 link: http://localhost:8180/training/update method: POST**
* **40 link: http://localhost:8180/training/update/confirm method: POST**
* **41 link: http://localhost:8180/unsigned/signatures/{id} method: GET**

Screenshot je urobený s <http://localhost:8180/homePageBackend>, obsahuje všetky url, na ktoré backend pocúva.

1,2 prijímajú „body“ a vracajú, ak v db nájdu json všetko o zamestnancovi okrem hesla.

1., prihlasovanie kartou (body: ‚card=?‘).

2., prihlasovanie cez login a heslo (post... body: ‚login=?&password=?‘).

3., 5., 6. ,8. ,9. vracajú podľa mena zoznam dvojíc jsonov [{meno, id},...], napr. [{department\_1,1}, {department\_2,2},...].

4., rušenie podpisov, očakáva v stringu zoznam id-éčiek(buď „5,12,13,1,4“ alebo 1 integer), ktoré má zrušiť, alebo „zrušiť zrušenie“, keď podpis nie je zrušený, tak ho zruší a naopak(využíva sa plpgsql funkcia).

7., je kontrolná url pre backend – kontrola, že stále beží

10., vráti pole dokumentov, ktoré nie sú editované a sú aktuálne(vráti json so všetkými atribútmi dokumentu + vypočítaniu mierou podpísania(complete): {„id“:..., „name“:..., „link“: ..., „note“:..., „type“:..., „release\_date“:..., „deadline“:.., „order\_number“:..., „version“:..., „prev\_version\_id“:..., „assigned\_to“:..., „require\_superior“:..., „edited“:..., „old“:..., + „complete“:...,}).

11., potvrdzuje jeden dokument, vyžaduje dosadiť namiesto „{id}“ jedno číslo(id dokumentu), priradí sa fieldu „edited“: false.

12., vytvorí v db dokument podľa jsonu, ktorý dostane od front endu vo formulári a zaradí ho medzi editované(json je ten istý ako v 10 bez: „edited“, „old“, „complete“, „prev\_version\_id“), ked dostane akékeľvek id!=0 tak v databáze vyhladá doc. S „id“ a nastaví „old“:true, presunie „id“ na „prev\_version\_id“ a vytvorí nový document s novým „id“ .

13., to isté ako 12, ale dokument hneď potvrdí.

14., vráti všetky editované(nepotvrdené dokumenty) s jsonom ako v 10 bez „complete“.

15., to isté ako 10 s tým, že očakáva json({„type“:..., „branch“:..., „city“:..., „department“:..., „division“:...,}) podľa ktorého vyfiltruje, musia to byť stringy, ktoré môže byť prázdne, môžu obsahovať jeden znak „x“(takýto znak sa používa aj pre nezaradené dokumenty), vtedy sa myslí, že nezáleží na tom fielde, alebo jedno číslo(id), alebo celé čísla oddelené čiarkami.

16. vráti pole „modifikovaných“ dokumentov(všetkých predchodcov dokumentu s daným „id“, ktoré majú všetko čo v 10(bez „complete“), ale majú vo vnútri „signatures“ – pole podpisov, ktoré majú v sebe(1:1) field „employee“:{...}.

17., json musí obsahovať id-dokum., urobí upgrade podľa id.

18., to isté ako v 17, navyše potvrdí.

19., vráti všetkých zamestnancov, ale iba niektoré údaje.

20. vráti všetkých, ktorý vyhovujú filtru zamestnancov, ale iba niektoré údaje, očakáva field „filter“ – je to string, ktorý sa bude posudzovať podľa regulár. výrazu(namiesto „filter“ je Query1):

where Query1::varchar similar to

'%(' || '[0-9,]\*' || e.branch\_id || '[0-9,]\*' || '|x); ' ||  
 '(' || '[0-9,]\*' || e.city\_id || '[0-9,]\*' || '|x); ' ||  
 '(' || '[0-9,]\*' || e.department\_id || '[0-9,]\*' || '|x); ' ||  
 '(' || '[0-9,]\*' || e.division\_id || '[0-9,]\*' || '|x)%'.

21., export podľa id nadriadeného a formátu. Jej princípom bolo vytiahnuť z db údaje a spustiť pythonovský skript(cmd+pipes), ktorý vyrobí .xslx súbor a po jeho ukončení poslať výsledok klientovi, v „skúšobnej“ prevádzke fungujúca doména, ale nepoužitá.

22., táto doméne očakáva .csv súbor, iný typ odmietne, k súboru očakáva značku importu, ak ju dostane pokladá súbor za import zamestnancov, ak nie, tak import kariet.

23., vracia konkrétny jazyk vo forme jsonu(stringu), namiesto {language} treba dosadiť skratku prichystaného jsona(SK alebo CZ) , v „skúšobnej“ prevádzke fungujúca doména(boli sme schopný dostať adekvátnu odpoveď), ale nepoužitá, okrem jazykov by mohla slúžiť na dodanie akéhokoľvek „ppriečinka.json“ vo form string, ktorý bude v priečinku „languages/scripts“(možno by sa v budúcnosti domény a aj package mohli premenovať a rozšíriť na „.../itemsAll“, „.../items/{dictionary}“ a „.../items/{dictionary}/{item}“.

24., vracia zoznam jazykov, ktoré sú k dispozícií(ich skratky) , v „skúšobnej“ prevádzke fungujúca doména, ale nepoužitá.

25., vráti zoznam stringov s menami všetkých lektorov.

26., očakáva string s id oddelenými čiarkami, kopíruje podpis a dá ho ako nepodpísaný(vytvorí nový), využíva plpgsql funkciu.

27., očakáva sa „id“ podpisu na dokument aby ho označila ako zamestnancom podpísaný.

28., to isté ako 27, len s nadriadeným.

29., to isté ako 27, len s podpisom na online training.

30., vráti všetky podpísané podpisy s dokumentami a s podradenými(v prípade nadriadeného) podľa id, json má štruktúru, ktorá obsahuje dve polia:

{ documents:[{dok.1, vlastnosť1,..., podpisy:[{...},...]}...], trainings:[{train.1, vlastnosť1,..., podpisy:[{...},...]}...]}.

31., vráti všetky(podpísané aj nepodpísané) podpisy od dokumentov podľa id chce vstupný json v tvare {meno: id}, pričom id je celé číslo a meno môže nadobúdať 4 hodnoty: „superior\_id", "employee\_id", "document\_id","filter", výsledný json má štruktúru: {documents:[{dok.1, vlastnosť1,..., podpisy:[{...},...]}...], employees:[{id, name,....}]}.

32., vráti všetky trainingy(potvrdené) – pole jsonov typu {„id“:.., „name“:.., „lector“:.., „agency“:.., „place“:.., „date“:.., „duration“:.., „agenda“:.., „deadline“:.., „employees“:.. }, pričom employees zodpovedá „unreleased\_id\_employees“ v databáze – toto slúži na menenie a pamätanie si, kto má podpísať ešte v nepodpísanom stave, v databáze je ešte field „edited“, ktorý sa ale cez json neposiela.

33., 34., 35., 39., 40., CRU(bez D) metódy +confirm podobne ako pri dokumentoch.

36., vráti všetky, ktoré sú stave edited, json je ten istý ako v 32.

37., vráti všetky podľa filtra, očakáva dáta(json[v js urlParams...]) dvoch stringov(jedného stringu alebo stringu a integer) a to {„lector“:..., „employee“:...} – pričom 1 so môže vynechať a „employee“ sa môže poslať ak ako integer.

38., vráti „modifikovaný“ training, v ktorom sú zahrnuté podpisy a ku podpisom 1:1 aj employee

41., to isté ako v 30, len namiesto podpísaných vráti nepodpísané.

Ešte pribudli:

**link: http://localhost:8180/document/manager/{id} method: GET**

zavádzajúci názov, vracia podobne ako v 10 podľa id, ale aj podľa id obyčajného zamestnanca

**link: http://localhost:8180/auth/logout method: GET**

ukončenie spojenia(maže internú regirtráciu)

**link: http://localhost:8180/employees/ids method: POST**

očakáva Json v tvare: {„ids“: „[1,2,3,........]“} a vráti first, last-name, id a anet\_id

**Frontend sa na servery spúšťa z staticky „zbuildovaných“ súborov, ktoré sa získajú za pomoci príkazu „npm run-script build „. Backend „obsluhuje“ frontend z priečinka build\_front\_end. V programe sú evidované sub-domény, na ktorých backend poskytuje prístup k frontendu:**

**http://localhost:8180/login**

**http://localhost:8180/records-to-sign**

**http://localhost:8180/signed-records**

**http://localhost:8180/add-record**

**http://localhost:8180/saved-record**

**http://localhost:8180/finder**

**http://localhost:8180/settings**

**http://localhost:8180/logout**

**http://localhost:8180/**

* všetky tieto domény obsluhuje jedna funkcia, ktorá „iba“ dáva prístup k statickým súborom.
* Každá nová doména frontendu musí byť „zaregistrovaná“ vo funkcii registerFrontend, ktorú nájdete v príslušnom package.