**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 1.16.2, 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%)

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 program-WARNING: v C-écku je Golangový garbage colektor vypnutý a samotné volania sú 10x 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 .txt súboroch, ktoré sa po spustení programu načítajú do pamäte(pokus urobiť SQL a kód nezávislými, ale kód potrebuje špecifické skripty, preto odporúčam ich brať ako celok), 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

1. **paths**

* Všetky mená subdomien, ktoré sa v programe vyskytujú, POZOR: bez metód a koncových znakov.
* 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.txt,end\_homepage.txt*)
* 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– mimo tento dokument)

* 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(*add\_sign\_after\_confirm\_doc.txt, confirm.txt*)
* Zobrazovanie podľa kategórií aktual, edited, filter
* Skripty k aktual, edited, filter (*aktual\_doc.txt, edited\_doc.txt, doc\_filter.txt*)
* 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(*filter\_employees.txt, all\_employees.txt*)
* Konfig na dovolenie prihlasovania bez hesla(*password\_allow.txt*)

1. **~~exports~~**

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

1. **~~languages~~**

* Nevyužitý package pre poskytovanie jazyka pre front end
* Iba na gite, branch „alternative\_language“

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(old\_document.txt, mail\_employee.txt, mail\_superior.txt)
* 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
* Dva pod-package:

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

* 2: scripts\_configs – Všetky skripty vyššie uvedených funkcii
* Využíva db funkciu: set\_sign\_new\_employees\_return\_emails
* Výnimka: confirm training sa robí tu!

1. **training**

* Vlastná štruktúra(trieda)
* Create, update
* Pohľad edited
* Skript (edited\_training.txt)

1. **upload\_export\_files**

* ~~Export využíva Python a dáva výsledok k dispozícií~~
* Import parsuje zamestnancov a kartičky z csv(kódovanie wndows1250) do db (pozn. iné odmieta)
* Využíva package signuture
* Pomocné skripty(*all\_employees\_from\_imports.txt, employees\_id\_anet\_id.txt, insert\_select\_id\_by\_name.txt*)
* Konfiguračný skript(*csv\_config.txt*) – nastavuje ktoré stĺpce sa majú považovať za príslušné údaje

**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.

2,3 prijímajú „body“ a vracajú, ak v db nájdu json všetko o zamestnancovi okrem hesla.

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

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

4.,6. ,8. ,9. vracajú podľa mena zoznam dvojíc jsonov [{meno, id},...], napr. [{department\_1,1}, {department\_2,2},...].

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

7., poskytuje všetky kombinácie, ktoré majú aktívny zamestnanci.

10., vráti pole dokumentov, ktoré nie sú editované a sú aktuálne.

11., potvrdzuje jeden dokument, vyžaduje dosadiť namiesto „{id}“ jedno číslo(id dokumentu).

12., vytvorí v db dokument podľa jsonu, ktorý dostane od front endu vo formulári a zaradí ho medzi editované.

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

14., vráti všetky editované(nepotvrdené dokumenty).

15., vráti všetky aktuálne dokum., ktoré nájde podľa filtra, ktorý dostane vo forme jsonu, v ňom sú 4 stringy(branches, ...), ktoré reprezentujú polia id, napr. {branches:“1,12,35,64,........“, cities:“5,50,...“, department:““, division:“1“}.

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

17., to isté ako v 16, navyše potvrdí.

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

19. vráti všetkých, ktorý vyhovujú filtru zamestnancov, ale iba niektoré údaje.

~~20., mala fungovať ako export podľa id nadriadeného a formátu, v „skúšobnej“ prevádzke fungujúca doména, ale nepoužitá. Jej princípom bolo vytiahnuť z db údaje a spustiť pythonovský skript(cmd+pipes), ktorý vyrobí .xslx súbor a po jeho ukončení poslať výsledok klientovi.~~

21., 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.

~~. na obr. Chýbajú vyhodené funkčné domény: „/languages/all“-GET, ktorá mala vracať zoznam jazykov, ktoré sú k dispozícií a „/language/{name}“-GET“, ktorá dáva konkrétny jazyk. Obe sú na gite.~~ ~~Bola by jediná doména, ktorá by čítala súbory za behu~~

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

23., očakáva sa id podpisu na dokument aby ho označila ako zamestnancom podpísaný.

24., to isté ako 23, len s nadriadeným.

25., to isté ako 23, len s podpisom na online training.

26., 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:[{...},...]}...]}.

27., vráti všetky(podpísané aj nepodpísané) podpisy od dokumentov podľa id, json má štruktúru: [{dok.1, vlastnosť1,..., podpisy:[{...},...]}...].

28., - 33., CRU(bez D) metódy +confirm podobne ako pri dokumentoch + view edited.

34. to isté ako 26, len s nepodpísanými podpismi.