1. JS

1.1. Front-end

1.1.1. initScene

súbor:

public/script.js

volanie:

initScene(lang)

vstupy:

lang:

JSON key values s textami pre jednotlivé komponenty

opis:

funkcia dostane zavolá funkciu na vyplnenie komponentov a nastaví počiatočnú veľkosť screenu a nastaví zavolá funkciu na nastavenie témy.

príklad:

```
# javascript
  eel.get_lang(lang)(initScene);
```

1.1.2. setLang

súbor:

public/script.js

volanie:

setLang(lang)

vstupy:

lang:

JSON key values s textami pre jednotlivé komponenty

opis:

funkcia dostane JSON, ktorý preiteruje podľa kľúčov a jednotlivým komponentom priradí hodnoty. Kľúče v JSONE sa musia zhodovať s id existujúcich komponentov.

1.1.3. setTheme

súbor:

public/script.js

volanie:

setTheme(theme)

vstupy:

theme:

light alebo dark

opis:

funkcia nastaví css súbor podľa toho akú tému máme zvolenú. Správne fungovanie danej funkcie zabezpečíme zavolaním funkcie changeTheme().

príklad:

```
# javascript
function changeTheme() {
    eel.get_conf("theme")(getThemeInHTML);
}

function getThemeInHTML(theme) {
    if (theme === "dark") setTheme("light");
    if (theme === "light") setTheme("dark");
}
```

1.1.4. clearContent

súbor:

public/script.js

volanie:

clearContent()

opis:

keďže aplikácia funguje pomocou komponentov v dive content, pri zobrazení nového komponentu je potrebné premazať to čo sa v ňom už nachádza. Funkcia teda prejde childy a zmení im display na none.

1.1.5. scrollFunction

súbor:

public/script.js

volanie:

table.parentElement.onscroll = function () {scrollFunction(table.parentElement, "samplesLabel");}

pozor:

pre naviazanie na novú tabuľku je potrebné pridať výpočet zmeny veľkosti do dictionary.

Každá tabuľka musí byť súčasťou div

vstupy:

tableClass:

referencia na div, v ktorom je tabuľka zaobalená

headingId:

ID nadpisu v danom stĺpci

opis:

Funkcia počíta posuny pri zmenšení nadpisov.

Objasnenie hodnôt premennej dict

95% - celková výška mínus 5% margin pre každý nadpis,

69px - výška nadpisu h2

34px - výška nadpisu h3

-16px-18px - odrátanie výšky elementu info a marginu tohto elementu

1.1.6. disableAllContent

súbor:

public/script.js

volanie:

disableAllContent()

opis:

vytvorí div, ktorý prekryje všetko okrem tlačidiel na zmenu témy, jazyka a načítanie vzoriek.

1.1.7. unDisableAllContent

súbor:

public/script.js

volanie:

unDisableAllContent()

opis:

opak k disableAllContent().

Table Selection Controller

1.1.8. addRowEventListener

súbor:

public/script.js

volanie:

tableSelectionController.addRowEventListener(tr, tableId);

vstupy:

tr:

riadkový element tabuľky s komplentnými dátami (už má aj všetky stĺpce)

tableId:

ID tabuľky, napríklad: "samplesTable"

opis:

funkcia priradí všetkým stĺpcom riadka listener. Po kliknutí sa zavolá funckcia selectRow()

1.1.9. selectRow

súbor:

public/script.js

volanie:

tableSelectionController.selectRow(row, tableId)

vstupy:

row:

riadkový element tabuľky (ten, na ktorý sa kliklo)

tableID:

ID tabuľky, napríklad: "samplesTable"

opis:

funkcia označí checkbox v tabuľke a zavolá funkciu markOne

1.1.10. markOne

súbor:

public/script.js

volanie:

tableSelectionController.markOne(tableId)

vstupy:

tableID:

ID tabuľky, napríklad: "samplesTable"

opis:

funkcia skontroluje či sú všetky checkboxy zaškrtnuté alebo nie a podľa toho, rozhodne o označení/odznačenní horného checkboxu.

1.1.11. markAll

súbor:

public/script.js

volanie:

tableSelectionController.markAll(tableId)

vstupy:

tableID:

ID tabuľky, napríklad: "samplesTable"

opis:

funkcia označí/odznačí všetky checkboxy podľa stavu horného checkboxu. Funkcia sa volá v index.html.

1.2. Back-end

1.2.1. saveImg

súbor:
public/script.js
volanie:
saveImg()

opis:
načíta sa cesta k obrázku a ten sa uloží.

1.2.2. saveExport

súbor:
public/script.js
volanie:
saveExport(all)

vstupy:
all:
boolean ci ukladáme všetko ale iba jeden súbor

opis:
ak je true hodnota all tak sa uložia všetky exporty, ak nie uloží sa konkrétny export.

1.2.3. getFiles

súbor:
public/script.js
volanie:
getFiles(files)

vstupy:
files:
JSON objekt so súbormi

opis:
funkcia iba prekonvertuje JSON z pythonu a pošle do funkcie na generovanie obsahu tabuliek.

1.2.4. loadTableData

súbor:

public/script.js

volanie:

loadTableData(obj)

vstupy:

obj:

prekonvertovaný JSON z pythonu do JSON objektu

opis:

funkcia vymaže obsah tabuliek, zobrazí tabuľky a zavolá dunkciu reloadTables na ich naplnenie.

1.2.5. reloadTables

súbor:

public/script.js

volanie:

reloadTables(obj)

vstupy:

obj:

prekonvertovaný JSON z pythonu do JSON objektu

opis:

funkcia vymaže obsah tabuliek, ktorý obsahuje a naplní ich novým obsahom.

1.2.6. getAllSelected

súbor:

public/script.js

volanie:

getAllSelected(table)

vstupy:

table:

element tabuľky, z ktorej chceme vybrať prvky.

výstupy:

data:

zoznam vzoriek, ktoré používateľ vybral

opis:

Funkcia prejde tabuľku, ktorú dostala na vstupe a vyberie vzorky, ktoré boli označené používateľom.

1.2.7. getAllSelectedModel

súbor:

public/script.js

volanie:

getAllSelectedModel(table)

vstupy:

table:

element tabuľky, z ktorej chceme vybrať prvky.

výstupy:

data:

zoznam modelov, ktoré používateľ vybral

opis:

Funkcia prejde tabuľku, ktorú dostala na vstupe a vyberie modely, ktoré boli označené používateľom.

1.2.8. parseLoadedData

súbor:

public/script.js

volanie:

parseLoadedData(data)

vstupy:

data:

JSON objekt, ktorý potrebuje spracovať

opis:

Funkcia rozbalí JSON dáta a následne ich opäť načíta do tabuľky.

1.2.9. draw

súbor:

public/script.js

volanie:

draw(data)

vstupy:

data:

zoznam informácii o obrázku

opis:

funkcia vyberie informácie o obrázku a vykreslí obrázok na plochu.

1.2.10. showResultsWithData

súbor:
public/script.js
volanie:
showResultsWithData(data)

vstupy:
data:
JSON objekt pre resultScreen

opis:
funkcia zobrazí potrebné komponenty a naplnia sa tabuľky so vzorkami.

1.2.11. showSampleDetail

súbor:
public/script.js
volanie:
showSampleDetail(element)

vstupy:
element:
riadok tabuľky, ktorý chceme zobraziť

opis:
podobné ako showResultsWithData akurát zobrazí detail vzorky.

1.2.12. showHomePage

súbor:
public/script.js
volanie:
showHomePage()

opis:
funkcia zobrazuje komponenty pre výber typu vzorky a nahranie zvolených vzoriek.

1.2.13. showVisualizationWithData

súbor:
public/script.js
volanie:
showVisualizationWithData(data)

vstupy:
data:

JSON objekt pre vizualizácia

opis:

podobné ako showResultsWithData akurát zobrazí vizualizáciu vzorky.

1.2.14. showModels

súbor:

public/script.js

volanie:

showModels()

opis:

funkcia zobrazuje komponenty pre vytvorenie nového modelov.

1.2.15. showClassification

súbor:

public/script.js

volanie:

showClassification(reload)

vstupy:

reload:

boolean, ktorý v sebe nesie informáciu, či je potrebné znovu načítať dáta v tabuľkách.

opis:

funkcia zobrazí komponenty pre klasifikáciu.

1.2.16. trainModels

súbor:

public/script.js

volanie:

trainModels()

opis:

funkcia ošetruje, či sú vybraté nejaké modely a vzorky, ak áno tak natrénuje nové vybrané modely na zvolených dátach.

1.2.17. deleteAllSelected

súbor:

public/script.js

volanie:

deleteAllSelected()

opis:

funkcia vymaže vybrané vzorky.

1.3. Aj pre python

1.3.1. showMessage

súbor:

public/script.js

volanie:

showMessage(data)

vstupy:

data:

súbor: {“message”: “”, “type”: 0}

type:

0 - error
1 - sucess
2 - warning
3 - hidden

opis:

funkcia vyparsuje z dát súbor a vloží do alertify podľa typu správy.

príklad:

```
# javascript
  eel.get_mess("noModelsOrSamples", 0)(showMessage);

# python
  eel.showMessage(get_mess("save", 1))
```

1.4. Alertify

Private object.

1.4.1. success, error, warning, hidden

súbor:

public/script.js

volanie:

alertify.success(message);
alertify.error(message);
alertify.warning(message);
alertify.hidden(message);

vstupy:

message:

String. Správa, ktorá sa zobrazí vo vyskakovacom okne.

wait:

Optional. Number. Počet milisekúnd, koľko má, čakať, kým sa okno zobrazí.

opis:

funkcia zobrazí alertify message

1.4.2. set

súbor:

public/script.js

volanie:

alertify.set({});

vstupy:

delay:

Počet milisekúnd, za koľko sa zobrazená správa skryje.

opis:

funkcia nastaví milisekundy, za koľko sa zobrazená správa skryje.

2. Python

2.1. get_lang

súbor:

ASR_app.py

volanie:

get_lang(language)

vstupy:

language:

zo languages.py sa podľa jazyka vyberu názvy k jednotlivim komponentom

výstupy:

súbor:

dict key value {'files': 'súbory'}

opis:

funkcia podľa jazyka vyberie JSON zo súboru lib/languages.py a pošle ho do aplikácie. Pri neznámom jazyku vrati eng a zaloguje chybu.

príklad:

```
# python
  get_lang(“sk”)

# javascript
  eel.get_lang(“it”);
```

2.2. get_mess

súbor:

ASR_app.py

volanie:

get_mess(code, type_mess)

vstupy:

code:

je key do súboru messge.py, odkiaľ sa vyberie hláška, ktorú chceme zobraziť

type_mess:

0 pre error a 1 pre succes

výstupy:

JSON:

{'message': ‘message’, 'type': type_mess}

opis:

funkcia podľa jazyka vyberie hlášku zo lib/message.py, tu pošle do js a k nej typ správy. Pri neznámom jazyku vráti eng a zaloguje chybu.

príklad:

```
# javascript
  eel.get_mess("noModelsOrSamples", 0);
```

2.3. get_data

súbor:

ASR_app.py

volanie:

get_data()

výstupy:

súbor:

dict key value {"files": samples_list, "models": models_list}

2.4. print_log

súbor:

ASR_app.py

volanie:

print_log(x)

vstupy:

x:

message log

príklad:

```
# python
  print_log(“messge”)
```

```
#javascript
  eel.print_log(“messge”);
```

2.5. select_files

súbor:

AS_app.py

volanie:

select_files(file_type, sample_type)

vstupy:

file_type:

typ súborov, ktoré chceme pridať, teda “models” alebo všeličo iné (pridajú sa vzorky)

sample_type:

typ vložených vzoriek (none, EVOO, LOO, VOO)

výstupy:

JSON:

{"files": samples_list, "models": models_list})

opis:

podľa typu sa spustí načítanie modelov alebo vzoriek.

2.6. save_file

súbor:

ASR_app.py

volanie:

save_file(url)

vstupy:

url:

cesta k obrázku, ktorý je zobrazený v aplikácii.

výstupy:

message o uložený súboru

2.7. save_export

súbor:

ASR_app.py

volanie:

save_export(sample)

vstupy:

sample:

názov vzorky, ktorú chceme exportovať, ak príde prázdny reťazec, uložia sa všetky.

výstupy:

message o úspešnom exporte

opis:

funkcia vytvorí txt súbor, s jednotlivými údajmi o klasifikovanej vzorke.

príklad:

```
# javascript
  save_export(“”);

  save_export(“sample.csv”);
```

2.8. classification

súbor:

ASR_app.py

volanie:

classification(JSON_data)

vstupy:

JSON_data:

models:

pole modelov

samples:

pole vzoriek

výstupy:

JSON:

pole vyhodnotení, jednotlivých vzoriek

[názov, názov súboru, výsledok, [názov modelu, výsledok modelu]]

opis:

funkcia vyberie vzorky a modely z JSON objektu, pošle ich do klasifikácie, ktorá vráti výsledky. Výsledky sa uložia do výstupného poľa.

príklad:

```
# javascript
  eel.classification({“samlpes”:[name, file_name], “models”:[model_file_name]});
```

2.9. training

súbor:

ASR_app.py

volanie:

training(JSON_data)

vstupy:

JSON_data:

models:

pole modelov

samples:

pole vzoriek

opis:

funkcia vyberie vzorky a modely z JSON objektu, natrénuje nové modely na vyraných dátach. Natrénovaný model sa uloží do priečinka k ostatným modelom.

príklad:

```
# javascript
  eel.training({“samlpes”:[name, file_name], “models”:[model_file_name]});
```

2.10. get_samples_list

súbor:

ASR_app.py

volanie:

get_samples_list( )

výstupy:

JSON:

pole vzoriek, ktoré sa globálne ukladajú počas behu

2.11. get_picture

súbor:

ASR_app.py

volanie:

get_picture(name, sample)

vstupy:

name:

názov vzorky, v podstate id

sample:

názov súboru

výstupy:

JSON:

name:

iba sa vráti name aké prišlo

sample:

názov súboru

path:

cesta k súboru od koreňového umiestnenia index.html

opis:

zavolá sa funkcia na vytvorenie obrázka, vrátia sa všetky potrebné informácie k zobrazeniu obrázka.

príklad:

```
# javascript
  eel.get_picture(name, sample);
```

2.12. get_conf

súbor:

ASR_app.py

volanie:

get_conf(key)

vstupy:

key:

hodnota kľúča, ktorú chceme vytiahnuť z conf

výstupy:

ak sa kľúč nájde v conf, vráti sa hodnota, ak nie tak sa vráti prázdny reťazec a zaloguje sa chyba.

príklad:

```
# python
  get_conf(“lang”)

# javascript
  eel.get_conf(“lang”);
```

2.13. set_conf

súbor:

ASR_app.py

volanie:

set_conf(key, value)

vstupy:

key:

kľúč k hodnote v conf

value:

hodnota ukrytá pod kľúčom

opis:

Ak potrebujeme prepísať conf, pošleme kľúč a hodnotu, funkcia si načíta celý conf a prepíše hodnotu, ktorú potrebuje, alebo pridá novú. Upravený conf sa opäť zapíše.

príklad:

```
# python
  set_conf(“lang”, “sk”)

# javascript
  eel.set_conf(“lang”, “sk”);
```

2.14. validate_and_get_class_from_csv

súbor:

ASR_app.py

volanie:

validate_and_get_class_from_csv(sample_path)

vstupy:

sample_path:

cesta k súboru

opis:

funkcia načíta scv do pandasu, pozrie či obsahuje hlavičku, ak áno vráti meno triedy, ak nie vrati -1.

2.15. write_samples(samples)

súbor:

ASR_app.py

volanie:

write_samples(sample)

vstupy:

sample:

kľúč k hodnote v conf

opis:

na vstupe dostane pole vzoriek, ktoré funkcia prejde a zapíše do sample_listu.

2.16. Loading screen

import:

from lib.loading_screen_controller import LoadingScreenPercentage

volanie:

lsp = LoadingScreenPercentage()

2.16.1. init_run

súbor:

lib/loading_screen_controller.py

volanie:

lsp.init_run(number_of_itterations)

vstupy:

number_of_itterations:

počet iterácií, ktoré má spraviť

opis:

inicializuje kruhový loading bar.

2.16.2. iterate_run

súbor:

lib/loading_screen_controller.py

volanie:

lsp.iterate_run()

opis:

funkcia zabezpečí iteráciu a včasné ukončenie runu. Beh ukončí po naplnení všetkých iterácií, ktoré boli na začiatku inicializované.

2.17. iter_lsp

súbor:

ASR_app.py

volanie:

iter_lsp

opis:

Zabezpečí volanie iterácie v súbore ASR_app.py. Keby sa funkcia volala tradičným spôsobom, vznikali by zacyklené importy.

3. Python lib

3.1. languages.py

súbor:

lib/languages.py

volanie:

svk_lang()
eng_lang()

výstupy:

JSON:

{key: value}

3.2. message.py

súbor:

lib/message.py

volanie:

svk_message()
eng_message()

výstupy:

dict:

{key: value}