Git - konvencie

Každý projekt bude vždy mať dve stabilne vetvy:
- main – reprezentuje stav PROD prostredia
- develop – reprezentuje stav DEV prostredia

Nové funkcionality (features) sú vždy vyvíjané v separátnych feature vetvách a nasledovné zlúčené do develop vetvy. Keď funkcionalita je pripravená na nasadenie develop vetva je zlúčená do main vetvy. Počas opravenia chýb na develop vetve sa vytvára nová bugfix vetva, ktorá po opravení bude zlúčená do develop vetvy. Počas opravenie chýb na main vetve sa vytvára nová hotfix vetva a po opravení chýb sa zlučuje s develop vetvou. Nasledovne je develop vetva zlúčená do main vetvy.

Všetky zlúčenia sú spravené cez Merge Request a pred potvrdením je potrebná review od iného z developerov.

Pomenovanie vetiev
Hlavná vetva (nasadzuje sa do PROD prostredia): main
Vývojová vetva (nasadzuje sa do DEV prostredia): develop
Feature vetva (dočasná vetva, môže byť vymazaná po zlúčení): feature/what_am_i_developing
Bugfux vetva (dočasná vetva, môže byť vymazaná po zlúčení): bugfix/what_am_i_fixing
Hotfux vetva (dočasná vetva, môže byť vymazaná po zlúčení): hotfix/what_am_i_fixing

Merge requests
Všetky zlúčenia sú vykonané pomocou Merge Request. Zlúčenie je vykonané žiadateľom Merge Request-u po akceptácii iným vývojárom. Akceptácia Merge Request-u je vykonaná po code review od iného vývojára a zapracovaní prípadných pripomienok tohto code review.

Konvencie písania kódu

Front End

Technológie
Ako Front End technológiu využívame **Quasar** založený na Vue.js (Javascript, CSS, HTML, Typescript)

Všeobecné pravidlá
- používame angličtinu - komentáre, názvy premenných, funkcií atď.
- na jeden súbor definujeme jeden komponent alebo službu
- snažíme sa písať krátke funkcie

Pomenovanie premenných, funkcií a tried

- Pre názvy funkcií a premenných sa používa výhradne camel case konvencia. Príklad: **camelCase**
- Na pomenovanie tried sa používa upper camel case konvencia. Príklad: **UpperCamelCase**
- uprednostňujeme jasný názov premennej alebo funkcie, pred krátkym nejasným názvom

Pomenovanie súborov

- používame konzistentné názvy súborov naprieč celým projektom
- názvy súborov vytvárame pomocou vzoru**: MainLayout.vue**
- slová v názve oddeľujeme vždy veľkým písmenom
- používame názvy, ktoré reprezentujú funkcionalitu (Layout, Store,...)

Komentáre

- Komentáre by sa mali používať vtedy, aby vysvetlili čokoľvek, čo môže byť na prvý pohľad nejasné.
- Komentáre v <template>
    - Jednoriadkové komentáre musia byť na jednom riadku a text vnútri musí byť ohraničený medzerami napr. <!-- Komentár -->
    - Viacriadkové komentáre musia začínať a končiť na vlastnom riadku a text nesmie byť odsadený. napr. <!-- (enter) Riadok číslo jedna (enter) Riadok číslo dva (enter) -->
- Komentáre v <script> a .ts súboroch
    - Štandardný JSDoc
    - Jednoriadkové komentáre začínajú s //.
    - Viacriadkové komentáre označujeme /* */.

Formátovanie kódu

- ESLint a Prettier- pokrýva problémy s kvalitou kódu a štýlom kódu

Quasar - základná štruktúra adresárov

```
.
public/
src/
        assets/
        components/         # .vue komponenty
        css/                # CSS/Sass/…
        layouts/            # layout .vue súbory
        pages/              # page .vue súbory
        boot/               # boot súbory (app initialization code)
        router/             # Vue Router
                index.js    # Vue Router - zadefinovanie
                routes.js   # App Routes - zadefinovanie
        store/              # Pinia Store
                index.js    # Pinia Store - zadefinovanie
```

```
        …
    stores/             # Pinia Store
        index.js        # Pinia inicializácia
        <store>         # Pinia stores...
        <store>...

    App.vue             # koreňový Vue komponent
    index.template.html # Template index.html
    quasar.config.js    # Quasar App Konfiguračný súbor
    babel.config.js     # Babeljs config
    .editorconfig       # editor config
    .eslintignore
    .eslintrc.js        # ESlint config
    .gitignore
    package.json        # npm scripts and dependencies
    README.md
```

Back End

Formátovanie

Napísaný Python kód musí zodpovedať štýlovým štandardom konvencie PEP 8 (https://peps.python.org/pep-0008/) a teda:

- Pre zarovnanie sa budú používať 4 medzerníky
- Limit pre maximálnu dĺžku riadku je 79 znakov
- Metódy mimo tried alebo definície tried sú oddelene dvoma prázdnymi riadkami, metódy vnútri tried sú oddelene jedným prázdnym riadkom
- Pri komentovaní riadkov alebo blokov kódu sa používa symbol #
- Pre rozsiahlejšie popísanie kódu alebo modulu sa používa dokumentovanie pomocou tz. docstrings, ktoré sú umiestnené medzi symbolmi """. Príklad komentáru: """Return an ex-parrot."""
- Pri menovaní tried sa používa CapitalizedWords konvencia. Príklad: NazovTriedy
- Pri menovaní metód sa používa lower_case_with_underscores konvencia. Príklad: nazov_metody

Pre jednoduchšie formátovanie kódu bude sa používať formátovací nástroj black (https://github.com/psf/black) v rámci tz. pre-commit hooks, ktorý formátuje kód v súlade s PEP 8 konvenciou. Takisto v rámci CI/CD prebehne krom na kontrolu správnosti formátovania kódu (???).

Komentovanie kódu

- Každý .py súbor musí obsahovať viacriadkový komentár s popisom komponentu, jeho funkcionality a účelu
- Každá metóda musí obsahovať viacriadkový komentár popisujúci vstupne parametre, výstup metódy, rady pri použití metódy a pod.

- Dôležite časti kódu vnútri metód musia obsahovať jednoriadkový komentár s popisom poskytujúcim lepšie pochopenie funkcionality

Testovanie

- Každá funkcionalita musí obsahovať svoj vlastný test
- Testy sú vykonávané pred commit-ovaniem zmien použitím nástroja pytest (https://docs.pytest.org/en/7.1.x/)

Logovanie

- Výsledky a medzivýsledky spracovaní musia byť zalogovane tak aby neunikli dôležite dáta v krokoch spracovania ale aj aby log nebol zahltený zbytočnými informáciami.
- Logovacia úroveň INFO sa používa pri oznámení dôležitých informácii napríklad začiatok alebo koniec operácii.
- Logovacia úroveň DEBUG sa používa na odhalenie chýb na testovacom alebo vývojovom prostredí.
- Logovacia úroveň ERROR sa používa pre oznámenie kritických chyb pri spracovaní dát.

README.md

Dokument README.md musí obsahovať:
1. Vysokoúrovňový popis funkcionalít aplikácie alebo komponentu
2. Sled spracovania (Processing pipeline) – koncove body a/alebo sled spracovania dát
3. Konfiguračne detaily – systémové premene ich účel a typ
4. Testovacie detaily – postup pri spustení jednotkových alebo integračných testov
5. Požiadavky – verzia Python ktorá sa používala, verzia Postresql a pod.

Dokumentácia Front Endu

Dokumentácia Front Endu:

- **export markdown/ html súborov z JSDoc komentárov z kódu**
  - JSDoc je open source generátor dokumentácie API pre Javascript.
  - Umožňuje vývojárom zdokumentovať svoj kód prostredníctvom komentárov.
  - Keď je váš kód úplne zdokumentovaný, môžete jednoducho vygenerovať webovú stránku obsahujúcu všetku dokumentáciu API spustením jednoduchého príkazu.
- **Vuepress dokumentácia** -
  - Popis dôležitých častí front end-u
  - Bol vytvorený na podporu dokumentačných potrieb vlastných projektov Vue.
  - Je generovaná statická stránka.
  - Každý súbor Markdown je skompilovaný do HTML (otvorí sa nové okno) a potom spracovaný ako šablóna komponentu Vue.

Dokumenty

Typy dokumentov a ich obsah tímu č. 12 zdieľajú rovnaké zaužívané zásady, ktoré sú nižšie popísané.
Písanie všetkých dokumentov je prostredníctvom nástroja Google Docs, ktoré medzi sebou členovia tímu zdieľame počas celej práce na projekte.

Štruktúrovanie dokumentov

Text dokumentov členíme podľa:

- nadpis 1. úrovne: Arial s veľkosťou písma 20 a farbou čierna
- nadpis 2. úrovne: Arial s veľkosťou písma 16 a farbou čierna
- nadpis 3. úrovne: Arial s veľkosťou písma 14 a farbou tmavosivá 4
- nadpis 4. úrovne: Arial s veľkosťou písma 12 a farbou tmavosivá 3
- normálny text: Arial s veľkosťou písma 11 a farbou čierna

Typy dokumentov

V tím č. 12 rozlišujeme tieto typy dokumentov, ktoré tvorí súbežne s vývojom webovej aplikácie.

Konvencie

Náš tím sa riadi týmito konvenciami, podľa ktorých boli vypracované tieto dokumenty:

- konvencie správy úloh
- konvencie písania kódu pre frontend a backend
- konvencie testovania a verziovania
- konvencie komunikácie
- konvencie písania dokumentov

Zápisnica

Každý zápis zo stretnutia obsahuje:

- dátum a čas
- miesto
- prítomní účastníci
- zapisovateľ zápisnice

V bodoch treba opísať aký bol priebeh stretnutie - čo sa preberalo. Okrem toho je pod tým samostatný odsek o tom, na akých úlohách, ktoré treba splniť, sme sa dohodli.

Retrospektíva

Dokument, ktorý vzniká ako záznam zo stretnutia tímu po každom šprinte, ktorý obsahuje, čo sa podarilo a čo sa nestihlo.

Dokumentácia

Súčasťou našej práce sú dva typy dokumentácie.

Dokumentácia inžinierskeho diela

Štruktúra dokumentácie:

1. "Big picture" - úvod, ciele a celkový pohľad na systém
2. Moduly systému
3. Technická dokumentácia

Dokumentácia riadenia projektu

Štruktúra dokumentácie:

1. "Big picture" - úvod, roly členov tímu a podiel práce, aplikácie manažmentov, sumarizácia šprintov, globálna retrospektíva
2. Motivačný dokument
3. Metodiky
4. Export evidencie úloh
5. Webové sídlo projektu