Dokumentácia k riadeniu

1. Úvod

Táto časť dokumentu popisuje riadenie tímu Predicar (v zoskupení tak, ako je popísané v kapitole sekcii 2) pri práci na projekte CarAd v rámci predmetu Tímový projekt. V rámci tejto časti ponúkneme opis rol členov tímu, aplikácie manažmentov, sumarizácie jednotlivých šprintov a globálnu retrospektívu za zimný semester.

Na túto časť nadväzujú všetky časti, ktoré súvisia s organizáciou práce v rámci školského projektu, ako sú nami používané metodiky a evidencie úloh jednotlivých šprintov.

2. Role členov tímu

Väčšinu kľúčových povinností sme si v rámci tímu rozdelili tak, aby každá pozícia bola obsadená aspoň dvoma členmi. Potom v prípade neprítomnosti niektorého člena môže projekt bezproblémovo napredovať.

Keďže si viacerí z nás chceli vyskúšať pozíciu Scrum mastera, striedame sa po jednom šprinte. Číslo šprintu, v ktorom sa daný člen tímu zhostil úlohy, je v zátvorke za menom pozície. Rozdelenie rol medzi členov nášho tímu je nasledovné:

- Bc. Baláž Branislav
    - Machine learning
    - Scrum master (3)
    - Frontend developer
    - Backend developer
- Bc. Bokor Peter
    - Machine learning
    - Project manager
    - Databázy
    - Backend developer
- Bc. Harnúšek Ondrej
    - NLP specialist
    - Backend developer
    - Tester
- Bc. Karel Max Dávid
    - Administrácia tímového servera
    - Git master
    - Scrum master (2, 5)
- Bc. Letanec Richard
    - Backend developer
    - Documentation Manager
- Bc. Vejčíková Veronika
    - NLP specialist

        - Správa tímového webu
        - Scrum master (1,4)
- Bc. Vrtal Tomáš
        - Frontend developer
        - Backend developer

3. Aplikácie manažmentov

Na efektívne riadenie projektu sme napísali nižšie popísané metodiky. Tými by sa mal riadiť každý člen tímu a ich kompletná forma sa nachádza v časti Metodiky.

3.1 Manažment komunikácie

Komunikácia je neoddeliteľnou súčasťou efektívneho riadenia tímu. Na komunikáciu používame výhradne Slack[1]. Tým zabezpečíme, že všetky informácie sú na jednom mieste aj spätne dohľadateľné. V Slacku máme vytvorené kanály, pričom každý má vlastný účel a pravidlá popísané v metodike komunikácie. Okrem toho máme integráciu s troma Slack botmi: Jira, Slack a Standup Alice. Posledný je na zefektívnenie standup meetingov v dňoch, keď sa nevieme stretnúť.

Výmenu dokumentov riešime cez tímový Google Drive. Popis jednotilivých kanálov a pravidiel komunikácie máme presne opísanú v Metodike komunikácie.

3.2 Manažment úloh

Na sledovanie stavu šprintu a jeho organizáciu používame nástroj Jira Software, ktorý máme nasadený na tímovom serveri. Každý člen tímu môže pridávať do backlogu *Tasky* alebo *Story*. Následne je na product ownerovi, aby sa staral o backlog a zoraďoval jednotlivé položky v backlogu podľa priority.

Ohodnotenie úloh sa deje priebežne metódou *planning poker*. Tým chceme docieliť, aby sme pri začínaní nového šprintu neminuli príliš veľa času na hodnotenie úloh, ale radšej rozhodli a diskutovali, čo má byť cieľom šprintu. Pred samotným hodnotením musí každá úloha obsahovať definition of done pre danú úlohu. Existencia jej podúloh nie je nutná.

Na záver šprintu je každá úloha odprezentovaná a product owner rozhoduje, či bude akceptovaná alebo nie.

3.3 Manažment verziovania

Pri spolupráci viacerých developerov je nutné efektívne zdieľať kód. Na to používame technológiu git a repozitáre máme hostované na github.com[2]. Rozhodli sme sa pre každú

[1] https://slack.com/intl/en-sk/
[2] https://github.com/

mikroslužbu vytvoriť vlastný repozitár, taktiež máme jeden pre tímovú stránku. Detaily a špecifikácie ako pracovať s touto technológiou máme v Metodike verziovania.

3.4 Manažment písania kódu

Na písanie kódu zatiaľ používame Python. Riadime sa štandardmi PEP (Python Enhancement Proposal) a používame typový Python. Tým chceme docieliť efektívnejšie prehliadky kódov a zlepšiť čitateľnosť kódov. Typový Python zas používame na detailnejšie použitie spomenutých pravidiel je popísané v Metodike písania kódu.

4. Sumarizácie šprintov

Šprint 1 - Benz

V tomto šprinte sme používali angličtinu na označenie úloh. Ukázalo sa, že to spôsobuje komplikácie a zdržania pri písaní definícií úloh. Preto sme neskôr prešli na Jiru v slovenčine.

Cieľ: To implement a prototype of the microservice architecture.

Trvanie: 11/Oct/19 11:24 AM - 24/Oct/19 3:20 PM

Názov úlohy	Popis	Zoznam podúloh	Odpracovaný čas	Akceptované
[CAR-6] Integrate Jira	As a team, we want to have integrated Jira software with our team website, in order to work efficiently and track our progress.	CAR-11	4h	Áno
[CAR-4] Create functional system prototype	As the product owner, I want a prototype of the system, to validate the proposed system architecture.	CAR-29, CAR-32, CAR-33, CAR-34, CAR-35, CAR-36, CAR-37	2d2h10m	Áno
[CAR-3] Submit TP Cup application	As a team, we want to attend TP Cup, in order to win.	CAR-7	1h30m	Áno

[CAR-2] Finalize team web site design	As a team, we need to have a website, to present our work.	CAR-8, CAR-9, CAR-10, CAR-28, CAR-30, CAR-31	7h15m	Áno
[CAR-1] Create 1st design of system architecture	As a product owner, I want to know the architecture of the system, to identify the components that will be used.	CAR-12, CAR-13, CAR-14, CAR-15, CAR-16	5h40m	Áno

Šprint 2 - Laurin & Klement

Cieľ: Chceme zjednodušit deployment a preskúmať ďalšie možnosti smerovania našej aplikácie z pohľadu používateľa

Trvanie: 24/Oct/19 5:20 PM - 07/Nov/19 3:33 PM

Názov úlohy	Popis	Zoznam podúloh	Odpracovaný čas	Akceptované
[CAR-44] Low fidelity user interface wireframe	As a user I want to see a low fidelity wireframe of a report, so I do know what to expect from the application.	CAR-54, CAR-55, CAR-56, CAR-57	1d0h35m	Áno
[CAR-43] Dockerise the application	As a programmer, I want to easily run up-to-date application with all components, so I can easily add new fancy features.	CAR-47, CAR-48, CAR-49, CAR-67	6h30m	Áno
[CAR-42] Create and connect with system	As a user, I want to be provided with reports based on long-term history, so	CAR-50, CAR-51, CAR-52,	2h30m	Áno

database prototype	reports are as precise as possible.	CAR-53		
[CAR-40] Create writing code in Python methodology	I want to know how to write and test code in Python.	CAR-60	4h	Áno
[CAR-39] Create code versioning methodology	I want to know how to create and name: commits, branches & pull requests		1h	Áno
[CAR-24] Integrate Jira and Slack	As a team, we want to attend TP Cup, in order to win.	CAR-27, CAR-45, CAR-46	1h	Áno

Šprint 3 - Fiat

Cieľ: Získavanie, predspracovanie a analýza dát.

Trvanie: 07/Nov/19 5:23 PM - 21/Nov/19 1:16 PM

Názov úlohy	Popis	Zoznam podúloh	Odpracovaný čas	Akceptované
[CAR-5] Analyze NLP tools for Slovak Language	Ako vlastník produktu chcem preskúmať možnosti automatického spracovania slovenského jazyka za účelom vytvorenia prototypu spracovania textu.	CAR-63, CAR-64, CAR-73	-	Áno
[CAR-68] Pridať github bota do Slacku	Ako tím chceme byť informovaní o udalostiach na našich repozitároch.	-	15m	Áno
[CAR-62]	Ako vlastník produktu	CAR-76,	-	Áno

Vytvorenie parsera pre inzeráty od Miroslava Ráca	chcem, aby boli spracované zozbierané inzeráty, aby sa dali ďalej spracovávať ako text. Potrebné je uloženie spracovaných dát, paralelizácia, naplnenie do db. Vysledne dáta obsahujú text inzerátu.	CAR-82, CAR-99		
[CAR-59] Extrakcia údajov z textu inzerátu - word2vec	Ako vlastník produktu chcem, aby bola možná extrakcia črty z inzerátu pomocou word2vec	CAR-81, CAR-83,	-	Áno
[CAR-75] Tool pre rýchle anotovanie a anotovanie dát	1000 anotovaných inzerátov.	CAR-80, CAR-84	-	Nie*
[CAR-71] Analyzovať portály inzerátov	Ako tím chceme poznať portály, z ktorých môžeme zbierať dáta.	CAR-77, CAR-78, CAR-88	-	Áno
[CAR-66] Skompletizovať veci na odovzdanie po 9. týždni	Pripravené veci na odovzdanie podľa pokynov na stránke.	CAR-74, CAR-85, CAR-86, CAR-87	-	Nie**
[CAR-58] Dátová analýza - dĺžka predaja auta	Ako používateľ chcem vedieť, ako dlho sa zvyknú predávať autá, aby som vedel robiť rozhodnutie pri nacenení.	CAR-79	-	Áno

* - CAR-59 - Nebol ukončený z dôvodu zlého nastavenia programu na anotáciu. Vzniknuté dáta boli nepoužiteľné.

** - CAR-66 - Vytvorené subtasky nedovoľovali ukončiť úlohu pred skončením šprintu. Konkrétne CAR-87 na odovzdanie dokumentov musí byť ůytvorený report z posledného šprintu.

Šprint 4 - Mercedes

Cieľ: Výsledkom šprintu je API, ktorá umožňuje prehliadať URL odkazy (a raw body) aktuálne platných inzerátov na predaj áut dostupných na slovenskom internete.

Trvanie: 21/Nov/19 4:31 PM - 05/Dec/19 3:13 PM

Názov úlohy	Popis	Zoznam podúloh	Odpracovaný čas	Akceptované
[CAR-102] Implementovať URL Updater	Potrebujeme mechanizmus, ktorý bude prechádzať databázu a vkladať odkazy do queue, aby sme vedeli, či je ešte inzerát platný.	CAR-116, CAR-125	-	Áno
[CAR-101] Vytvoriť API, ktorá bude z DB vracať inzeráty	Nech vracia len aktuálne inzeráty.	CAR-112, CAR-113	-	Áno
[CAR-100] Vytvorenie crawlera a generátora	Ako product owner chcem mať crawler, ktorý bude z queue získavať URL, ktoré má stiahnuť, a asynchrónne vytvárať callbacky na spracovanie stiahnutého HTML. Generátor bude vytvárať a postupne vyberať záznamy z queue.	CAR-103, CAR-104, CAR-105, CAR-106, CAR-107, CAR-114	-	Áno

[CAR-66] Skompletizovať veci na odovzdanie po 9. týždni	Pripravené veci na odovzdanie podľa pokynov na stránke.	CAR-74, CAR-85, CAR-86, CAR-87	-	Áno
[CAR-98] Implementovať queue pre URL inzerátov	Ako product owner chcem mať queue, ktorá bude spoĺňať: - collection v MongoDB - riešenie duplicitných	CAR-108, CAR-109, CAR-110, CAR-111, CAR-122	-	Nie*
[CAR-89] Parsovanie inzerátov z auto-moto portálov	Ako používateľ chcem mať dostupné dáta z čo najviac inzerátov na slovenskom internete, aby boli štatistiky a predikcie relevantné.	CAR-91, CAR-92, CAR-93, CAR-94, CAR-95, CAR-115, CAR-117, CAR-124	-	Nie**
[CAR-75] Tool pre rýchle anotovanie a anotovanie dat	1000 anotovaných inzerátov.	CAR-80, CAR-84	-	Nie***

* - CAR-98 - Nebolo včas dokončené generovanie odkazov na indexové stránky.

** - CAR-89 - Vyskytol sa technický problém, ktorý skomplikoval splnenie úlohy.

*** - CAR-75 - Nebola splnená základná podmienka podľa definition of done.

Šprint 5 - Rolls Royce

Cieľ: Výsledkom šprintu je API na predikciu ceny podľa URL.

Trvanie: 05/Dec/19 5:02 PM - 12/Dec/19 5:02 PM

Názov úlohy	Popis	Zoznam podúloh	Odpracovaný čas	Akceptované
[CAR-98] Implementovať queue pre URL inzerátov	Ako product owner chcem mať queue, ktorá bude spĺňať: - collection v MongoDB	CAR-108, CAR-109, CAR-110, CAR-111, CAR-122	-	Áno

	- riešenie duplicitných záznamov v queue			
[CAR-128] Pripraviť dokumentáciu na odovzdanie po zimnom semestri	Ako tím chceme mať dokončenú dokumentáciu všetkých častí potrebných na úspešné ukončenie predmetu Tímový projekt I. To nezahŕňa časti, ktoré musia byť zapísané až po dokončení šprintu.	CAR-129, CAR-130, CAR-132, CAR-133	-	Áno
[CAR-131] Napísať metodiku dokumentácie	-	-	-	Áno
[CAR-134] Upraviť predikčnú funkciu na prediktore, aby vracala predikciu na základe features pre stiahnuté inzeráty	Predikcia bude fungovať len pre bazáre, kde existujú v databáze vyparsované features	CAR-135, CAR-136, CAR-137, CAR-138	-	Áno

5. Globálna retrospektíva za ZS

Čo bolo dobré?

V rámci sme dokázali už pred začatím práce na projekte rýchlo pripraviť všetky potrebné komunikačné prostriedky. Súčasťou tohto procesu bolo aj definovanie komunikačných kanálov v aplikácii Slack, aby zostali informácie, ktoré si posielame vždy prehľadné.

O trochu neskôr sme tiež zaviedli kanál určený pre standup, pomocou ktorého máme lepší kontakt s postupom práce ostatných v tíme. Časom sme sa tiež naučili dodržiavať pravidlo o povinnej účasti na standup-och v stredu a nedeľu.

Okrem toho sme rýchlo zostavili metodiky, o ktoré sa môžeme pri práci oprieť.

Čo je potrebné zlepšiť?

Počas prvých šprintov sa stávalo, že sme dorábali veci tesne pred koncom šprintu, kvôli čomu nám zostávalo málo času na dôkladné review. Čakanie na review bolo tiež vo veľa prípadoch dlhšie, než by bolo vhodné.

Na štvrtkových stretnutiach, hlavne pri plánovaní, nám často zostalo málo času, čoho následkom boli nedostatky v plánovaní a nedorozumenia v popisoch úloh, ktoré sme si neskôr museli vysvetľovať.

V 4. šprinte sme mali naplánované viaceré úlohy, ktoré spočívali v rozšírení kódu pre zber dát. Veľa týchto úloh bolo úzko previazaných a rozhodnutie pri plnení jednej mohlo požadovať zmenu aj v úlohách, ktoré sme už považovali za dokončené. Keď tento prípad nastal nám chýbali predbežné dohody a nemali sme definované ideálne poradie, v ktorom by tieto úlohy mali byť riešené.

Globálny cieľ na ZS a jeho splnenie

“**Chceme vidieť vypočítanú predikciu ako odhad ceny v našom rozhraní pre štandardný vstup.**”

Cieľ, ktorý sme si stanovili pred začatím prvého šprintu v plánovacej fáze, sa nám podarilo vo všetkých podstatných častiach splniť. Nastavenie prostredia a pospájanie všetkých stavebných blokov systému, ktoré si tento cieľ vyžadoval, si však vyžiadali množstvo času a predikcia, ktorú náš systém generuje zatiaľ nie je výstupom nejakého sofistikovanejšieho prístupu, pri ktorom by sme sa v budúcnosti plánovali zostať.

Metodiky

Metodika komunikácie

1. Dedikácia

Metodika definuje princípy a pravidlá komunikácie pre tím Predicar. Opisuje komunikáciu medzi členmi tímu, medzi členmi tímu a vedúcim tímu a komunikáciu s verejnosťou.

2. Osobné stretnutia

Najväčšia časť komunikácie prebieha na spoločných stretnutiach tímu. Formálne stretnutie tímu aj s vedúcim prebieha v štvrtok od 14:00 do 17:00, kde sa hlavne diskutuje o projekte a jeho vývoji.

Okrem toho sa tím stretáva aj v pondelok od 14:00 do 18:00, kde okrem diskusie tím spoločne pracuje na projekte.

3. Slack

Na spoločnú komunikáciu ak nie je možnosť osobného stretnutia využívame nástroj Slack. Každý člen tímu musí pravidelne kontrolovať Slack, či už na mobile alebo počítači - teda každý deň v prípade potreby komunikovať. V tomto nástroji máme viacero komunikačných kanálov:

- **brainstorming** - Kanál, ktorý slúži na zdieľanie akýchkoľvek nápadov či pripomienok.
- **general** - Kanál, ktorý slúži na všeobecnú diskusiu o projekte a úlohách a taktiež aj v prípade, že člen tímu potrebuje radu alebo pomoc.
- **jira** - Kanál, ktorý slúži na notifikácie z nástroja Jira. Nachádzajú sa tu správy o vytvorení, editovaní, priradení a komentovaní príbehu alebo úlohy a ešte správy o tom ak úloha sa dostane do stavu “Ready for Review” a “Done”, čo nám slúži na lepší prehľad o tom čo kto robí.
- **methodics** - Kanál, ktorý slúži na komunikáciu o upravovaní existujúcich a tvorení nových metodík.
- **random** - Kanál, ktorý slúži na komunikáciu netýkajúcu sa tímového projektu.
- **standup** - Kanál, na ktorom každú stredu a nedeľu večer do 20:00 každý člen tímu napíše, na čom aktuálne pracuje a stav úlohy. Na pripomínanie nám slúži Standup Alice bot, ktorý každý deň vyzve každého člena tímu aby vyplnil formulár o jeho činnosti, s tým že v ostatné dni okrem stredy a nedele sa nemusí vyjadriť.
- **tim-web** - Kanál, na ktorom sa riešia všetky veci týkajúceho sa webovej stránky tímu.
- **tp_cup** - Kanál, na ktorom sa rieši všetko súvisiace s TP Cup-om.
- **uzitocne_linky** - Kanál, kde sa pridávajú užitočné odkazy aj s popisom aby sa dali jednoducho kedykoľvek dohľadať.

Okrem týchto kanálov, môže člen využiť priamu komunikáciu s iným členom tímu v súkromnom chate.

4. Email

Náš tím používa spoločný e-mail pre komunikáciu s verejnosťou, odkiaľ sa preposielajú e-mailové správy všetkým členom tímu - **fiit.tp.22@gmail.com**. Prístup k tomu mailu majú všetci členovia tímu okrem vedúceho tímu.

Metodika verziovania kódu

1. Dedikácia

Metodika definuje princípy a pravidlá uchovávania a verziovania zdrojového kódu pre tím Predicar. Opisuje použité technológie a spôsob tvorby commitov, vetiev a pull requestov.

2. Použité technológie

Na uchovávanie a verziovanie kódu používame webovú službu Github podporujúcu vývoj softvéru s pomocou verziovacieho nástroja Git.

Na Githube máme vytvorenú organizáciu https://github.com/predicar, ktorá obsahuje všetky repozitáre vytvorené našim tímom.

3. Commit

Pomenovaná množina zmien v kóde.

Pravidlá pre vytváranie commitov:

- Commit správa začína vždy slovesom: Add, Fix, Remove, Change, Revert
- Commit správa musí byť pochopiteľná aj bez znalosti daného programovacieho jazyka
- Commit správy sa píšu výlučne v angličtine
- Commitovať treba ucelené, ideálne aj funkčné celky práce
- Jeden commit pokrýva práve jednu zmysluplnú zmenu v kóde
- Zmeny vo formátovaní kódu sú commitované samostatne
- Dĺžka commit správy by nemala presiahnuť viditeľnú dĺžku
- Commit nesmie pokrývať skryté zmeny, ktoré nie sú opísané v správe
- Ak commit súvisí s github issue treba v správe commitu spomenúť vo formáte “#<číslo issue>”
- Ak commit opravuje pripomienky z code review treba v commit správe vo formáte “#<číslo pull requestu>”
- Obsahuje referenciu na Jira task

4. Vetva

Pomenovaná množina commitov určená pre implementáciu konkrétnej funkcionality.

Rozdelenie vetiev:

- master
    - hlavná vetva, ktorá odráža aktuálny stav na produkčnom serveri
    - je zamknutá na priame pushovanie do vetvy, povolené len cez pull requesty z dev vetvy
- dev
    - vývojová vetva, ktorá odráža aktuálny stav na testovacom serveri
    - je zamknutá na priame pushovanie do vetvy, povolené len cez pull requesty z ďalších vetiev
- ďalšie vetvy:
    - vetvy pre implementáciu novej funkcionality:
        - následne sa zlučujú s vetvou dev pomocou pull requestu
        - názov vetvy je Jira issue ID s predponou “feature” (napr. *feature/CAR-1*)

        - pre každú Jira issue je vytvorená maximálne jedna vetva v danom repozitári
        - ak robia viacerí na jednej Jira issue, každý má vytvorenú vlastnú a potom sa merguje. Názov takejto vetvy je napr feature/CAR-1/meno
    - vetvy pre rýchlu opravu chýb:
        - používajú sa len v prípade, že je nevyhnutná okamžitá zmena kódu na produkcii
        - následne sa zlučujú s vetvami master a dev pomocou pull requestu
        - názov vetvy je stručné pomenovanie opravy s predponou “hotfix” (napr. *hotfix/update-db-credencials*)

5. Pull request

Žiadosť o presunutie implementovanej funkcionality z jednej vetvy do inej.

Pravidlá pre vytváranie pull requestov:

- Názov pull requestu je rovnaký ako ID a názov príslušného Jira issue (napr. *CAR-1 Vytvorenie 1. dizajnu systémovej architektúry*)
- Opis pull requestu obsahuje doplňujúce informácie dôležité najmä pri code review
- Názov a popis pull requestu sa píše výlučne v slovenčine
- Každá vetva môže mať maximálne jeden schválený pull request
- Po schválení a zlúčení vetvy sa vetva vymaže (pokiaľ nejde o vetvu dev)

Metodika písaniu kódu v jazyku Python

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one—and preferably only one—obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than right now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea—let's do more of those!

The Zen of Python (PEP 20)

1. Dedikácia

Metodika definuje princípy a pravidlá písania kódu v jazyku Python. Opisuje štýl písania, štandardy ktoré používame a dokumentáciu kódu.

2. Zaužívané štandardy

Pri písaní kódu sa riadime zaužívanými štandardmi PEP 8 (Style Guide for Python Code) pre písanie kódu, PEP 257 (Docstring Conventions) pre dokumentáciu kódu a PEP 484 (Type Hints) pre typovanie kódu.

Kód píšeme v anglickom jazyku, premenným, metódam a triedam dávame výstižný názov.

Názvy premenných a metód píšeme celé malými písmenami, jednotlivé slová oddelené podtržníkom (_).

Názvy tried píšeme štýlom UpperCaseCamelCase.

Funkcie a definície tried sa oddeľujú 2 prázdnymi riadkami. Definície metód v triedach sa oddeľujú 1 prázdnym riadkom. Na konci súboru musí byť 1 prázdny riadok.

```
class ExampleClass():

        def example_method():

        example_var1 = 1
        example_var2 = 2

        return example_var1 + example_var2
```

3. Používanie nápovedí typov (type hints)

Pri písaní kódu používame nápovede typov premenných dovážaných do funkcií a typov vracaných funkciami.

```
def example_method(number: int, letter: String) -> String:

        return letter + str(number)
```

4. Dokumentovanie kódu

Komentáre, podobne ako kód, píšeme v **anglickom** jazyku. Riadime sa pri tom štandardom PEP8. Blokové komentáre platia pre kód pod nimi a sú odsadené na rovnakej úrovni ako kód, ktorý komentujú. Každý riadok blokového komentáru začína znakom **#** a jednou medzerou.

Pri písaní docstringu pre metódy a triedy sa riadime štandardom PEP257 pre docstring konvencie a používame Google štýl písania docstringov. Každý docstring obsahuje krátky a výstižný opis metódy, alebo triedy. Každý docstring opisuje privezené argumenty ako Args, typ návratu ako Returns a vyvolané výnimky ako Raises.

```
def connect_to_next_port(self, minimum):
        """Connects to the next available port.

        Args:
        Minimum(int): A port value greater or equal to 1024.

        Returns:
        int: The new minimum port.

        Raises:
        ConnectionError: If no available port is found.
        """
  if minimum < 1024:
        # Note that this raising of ValueError is not mentioned in the doc
        # string's "Raises:" section because it is not appropriate to
        # guarantee this specific behavioral reaction to API misuse.
        raise ValueError('Minimum port must be at least 1024, not %d.' % (minimum,))
```

```
port = self._find_next_open_port(minimum)
if not port:
        raise ConnectionError('Could not connect to service on %d or higher.' % (minimum,))
assert port >= minimum, 'Unexpected port %d when minimum was %d.' % (port, minimum)
return port
```

Viac o Google štýle písania docstringov:

- https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html
- http://google.github.io/styleguide/pyguide.html

Metodika dokumentácie

1. Dedikácia

Metodika opisuje proces riadenia dokumentácie pre tím Predicar. Opisuje konkrétne dokumenty a ich štruktúry, ktoré by mali obsahovať.

2. Dokumenty

Pre riadenie dokumentácie v tíme je potrebné zapisovať nasledujúce dokumenty:

- Inžiniersko dielo - Big Picture
- Riadenie - Big Picture
- Moduly systému
- Inštalačná príručka
- Používateľská príručka
- Technická dokumentácia
- Metodika(y)

Dokumentácia inžinierskeho diela - Big Picture

Štruktúra dokumentácie inžinierskeho diela

- Úvod
- Ciele projektu
- Celkový pohľad na systém

Štruktúra dokumentácie modulov systému

- Analýza
- Návrh
- Implementácia
- Testovanie

Dokumentácia riadenia - Big Picture

Štruktúra dokumentu Riadenia:

- Úvod
- Role členov tímu

- Podiel práce
- Aplikácie manažmentov
- Sumarizácie šprintov
- Globálnu retrospektívu šprintov

Štruktúra sumarizácie šprintu

- Názov šprintu
- Cieľ šprintu
- Trvanie šprintu (od-do)
- Tabuľka s príbehmi a úlohami, s popisom, odpracovaným časom a indikáciou akceptovania

Dokument metodiky

Štruktúra metodiky

- Titulná strana s názvom metodiky
- Dedikácia
- Roly - v prípade potreby
- Sekcie metodiky

3. Text

Text v dokumentoch by mal mať nasledovné parametre:

- Názov - Arial s veľkosťou 26
- Nadpis 1 - Arial s veľkosťou 20
- Nadpis 2 - Arial s veľkosťou 16
- Nadpis 3 - Arial s veľkosťou 14
- Text - Arial s veľkosťou 11