Paca

Grupp 5

Kodgranskning

V. 0.6

12 april 2018

Dokumenthistorik

Datum	Version	Beskrivning	Författare
180313	0.1	Påbörjade file naming conventions och byggde upp strukturen för riktlinjer för kod.	Milo Bengtsson
180404	0.2	Syfte	Milo Bengtsson
180410	0.3	Inledning (syfte, ord + referenser som kommer användas i texten), påbörjade struktur i riktlinjer för kod, struktur för hela dokumentet	Milo Bengtsson
180411	0.4	2.1 och 2.1.1	Milo Bengtsson
180412	0.5	2.1.3 och 2.1.4	Milo Bengtsson
180413	0.6	Skrev metod, process och sammanställde ordlista och referenser.	Milo Bengtsson
180413	1.0	Första färdiga versionen lämnas in på It's Learning inför RS2.	Milo Bengtsson

Innehåll

1. Inledning

Detta dokument är en del av den individuella fördjupningen kring kodgranskning, vilket projektmedlemmen Milo Bengtsson ansvarar för.

1.1 Syfte

Syftet med kodgranskningsdokumentets första del är att skapa riktlinjer för kodskrivning, namngivning och filstruktur för att få en så konsekvent kod och struktur som möjligt. Dessutom tillåter det alla gruppmedlemmar i det egna projektet, såväl som utomstående, att förstå och kunna följa med i dokumenten eftersom alla följer samma konventioner.

Kodgranskningsdokumentets andra del syftar till att standardisera hur projektets kod granskas och testas. Genom att använda en och samma checklista för all typ av kodgranskning, samt följa ett definierat protokoll för mötena kring det, undviker gruppen inkonsekventa mätverktyg och irrelevanta resultat. Utöver att vara ett verktyg för att upptäcka och förebygga fel och brister i koden, fungerar kodgranskningen även som ett lärotillfälle för de projektmedlemmar som inte varit lika involverade i det granskade programspråket.

1.2 Ordlista

Ord	**Förklaring**
CSS	*Eng. Cascading Style Sheets*. Ett stildokument som beskriver hur ett HTML-dokument ska presenteras rent visuellt på en sida.
deklaration	Benämning för den text inom måsvingar som anger hur en viss selektor (t.ex. element, klass, ID) ska formateras.

Django	Ett webbramverk för utveckling av webbsidor i programmeringsspråket Python.
docstring	*Eng. document string*. En textsträng som förekommer som det första uttalandet i en modul, funktion, klass eller metoddefinition.
HTML	*Eng. HyperText Markup Language.* Den grundläggande standarden för hur webbsidor skrivs i dokumentform.
peer-review	*Eng. referentgranskning*. Ett givet dokument granskas av såväl utvecklare som gruppmedlemmar, i syfte att på ett mer objektivt sätt hitta brister och fel i koden.
PEP8	*Eng. Python Enhancement Proposal 8*. En standard och stilguide för hur Pythonkod bör skrivas.
preprocessor	Ett program som behandlar sin inputdata för att ge outputdata som används som input till ett annat program.
Sass	*Eng. Syntactically Awesome Style Sheets.* En förlängning av CSS som tillåter användningen mer avancerade funktioner.
template language	*Eng. mallspråk*. Ett programspråk är inbäddat i ett annat, och tillåter platshållare i som senare dynamiskt byts ut baserat på vilken data som skickas till dokumentet.
UTF-8	Teckenkodning som används för att representera text kodad i Unicode.
W3C	*Eng. World Wide Web Consortium*. En internationell sammanslutning av representanter inom webb- och IT-branschen som utvecklar standarder för att säkerställa långsiktig utveckling av webben.
WCAG	*Eng. Web Content Accessibility Guidelines*. En av W3C etablerad uppsättning rekommendationer för tillgängligt innehåll på webben.

1.3 Referenser

[1] Django Software Foundation, “Django Documentation”, *Django*, u.å. [Online]. Tillgänglig: https://docs.djangoproject.com/en/2.0/. [Hämtad: 11 april 2018].

[2] Python Software Foundation, “PEP 8 -- Style Guide for Python Code”, *Python*, 1 augusti 2013. [Online] Tillgänglig: https://www.python.org/dev/peps/pep-0008/. [Hämtad: 10 april 2018]

[3] The PostgreSQL Global Development Group, “PostgreSQL: The world's most advanced open source database”, *PostgreSQL*, u.å. [Online]. Tillgänglig: https://www.postgresql.org/. [Hämtad: 13 april 2018].

[4] The PostgreSQL Global Development Group, “PostgreSQL 10.3 Documentation”, *PostgreSQL*, u.å. [Online]. Tillgänglig: https://www.postgresql.org/docs/10/static/index.html. [Hämtad: 13 april 2018].

[5] S. Holywell, “SQL Style Guide”, *SQL Style Guide*, u.å. [Online]. Tillgänglig: http://www.sqlstyle.guide/. [Hämtad: 13 april 2018].

[6] C. Peters, “3 Key Software Principles You Must Understand”, *Code Envato Tuts+*, 7 september 2012. [Online]. Tillgänglig: https://code.tutsplus.com/tutorials/3-key-software-principles-you-must-understand--net-25161. [Hämtad: 12 april 2018].

[7] S. Faulkner et al., “HTML 5.2”, *World Wide Web Consortium*, 14 december 2018. [Online]. Tillgänglig: https://www.w3.org/TR/2017/REC-html52-20171214/. [Hämtad: 11 april 2018].

[8] B. Caldwell et al., “Web Content Accessibility Guidelines (WCAG) 2.1”, *World Wide Web Consortium*, 30 januari 2018. [Online]. Tillgänglig: https://www.w3.org/TR/WCAG21/. [Hämtad: 11 april 2018].

[9] Sass, “Sass (Syntactically Awesome StyleSheets)”, *Sass: Syntactically Awesome Style Sheets*, 2 april 2018. [Online] Tillgänglig: https://sass-lang.com/documentation/file.SASS_REFERENCE.html. [Hämtad: 10 april 2018]

[10] T. Atkins Jr. et al., “CSS Snapshot 2017”, *World Wide Web Consortium*, 31 januari 2018. [Online]. Tillgänglig: https://www.w3.org/TR/CSS/. [Hämtad: 11 april 2018].

[11] T. Çelik et al., “Selectors Level 3”, *World Wide Web Consortium*, 30 januari 2018. [Online]. Tillgänglig: https://www.w3.org/TR/selectors-3/. [Hämtad: 11 april].

[12] J. Daggett et al., “CSS Fonts Module Level 3”, *World Wide Web Consortium*, 15 mars 2018. [Online]. Tillgänglig: https://www.w3.org/TR/css-fonts-3/. [Hämtad: 11 april 2018].

[13] B. Bos et al., “CSS Backgrounds and Borders Module Level 3”, *World Wide Web Consortium*, 17 oktober 2017. [Online]. Tillgänglig: https://www.w3.org/TR/css-backgrounds-3/. [Hämtad: 11 april 2018].

[14] T. Çelik et al., “CSS Color Module Level 3”, *World Wide Web Consortium*, 15 mars 2018. [Online]. Tillgänglig: https://www.w3.org/TR/css-color-3/. [Hämtad: 11 april 2018].

[15] M. Ricalde, “Nested selectors: the inception rule”, *The Sass Way*, 20 november 2011. [Online]. Tillgänglig: http://thesassway.com/beginner/the-inception-rule. [Hämtad: 12 april 2018]

[16] The jQuery Foundation, “jQuery: The Write Less, Do More, JavaScript Library”, *jQuery*, 2018. [Online]. Tillgänglig: https://jquery.com/. [Hämtad: 13 april 2018].

[17] FullCalendar LLC, “FullCalendar - JavaScript Event Calendar”, *FullCalendar*, 2018. [Online]. Tillgänglig: https://fullcalendar.io/. [Hämtad: 13 april 2018].

[18] B. Terlson, “ECMAScript® 2019 Language Specification”, *ECMA International, Technical Committee 39 - ECMAScript*, 21 mars 2018. [Online]. Tillgänglig: https://tc39.github.io/ecma262/. [Hämtad: 11 april 2018].

[19] F. Aboukhadijeh, “Rules”, *JavaScript Standard Style*, u.å. [Online]. Tillgänglig: https://standardjs.com/rules.html. [Hämtad: 13 april 2018].

[20] F. Tsui, O. Karam och B. Bernal, *Essentials of Software Engineering*, 3 uppl. Burlington: Jones and Bartlett Learning, 2014.

[21] I. Sommerville, *Software Engineering*, 9 uppl. Boston: Addison-Wesley, 2011.

[22] V. Bryukhanov, *PEP8 online*, u.å. [Online]. Tillgänglig: http://pep8online.com/. [Hämtad: 11 april 2018].

[23] World Wide Web Consortium, *Markup Validation Service*, u.å. [Online]. Tillgänglig: https://validator.w3.org/. [Hämtad: 11 april 2018].

[24] World Wide Web Consortium, *CSS Validation Service*, u.å. [Online]. Tillgänglig: http://jigsaw.w3.org/css-validator/. [Hämtad: 11 april 2018].

2. Metod

2.1 Riktlinjer för kod

Här listas de riktlinjer som ska användas av gruppen för den kod som skrivs under projektet, indelat i programspråk. Dessa inleds med bestämmelser kring namngivning av filer, eftersom det förkortar tid det tar att läsa och förstå vad varje fil innebär och innehåller. Vidare standardiseras namngivning av diverse beståndsdelar som förekommer i de olika språken, för att på så sätt kunna fokusera på viktigare uppgifter än att diskutera meningsskiljaktigheter kring och korrigera syntax och namnstandarder.

2.1.1 Generella konventioner

- Filnamn ska endast innehålla gemener.
- Om ett filnamn består av flera ord som är enkla att urskilja så skrivs dessa ihop utan specialtecken som avdelare, t.ex. `scrollpsy.js`.
- Filnamn, kod och kommentarer i kod ska skrivas på engelska.
- Teckenkodning ska vara *UTF-8*.
- Ingen blandning av programspråk.
    - Undantag *Django Template Language*, som kombinerar *HTML* och Python.
- All indentering ska bestå av en tabb, vilket motsvarar fyra mellanslag.
- Namngivning ska vara meningsbärande, det vill säga att exempelvis klasser, funktioner och ID ska ha deskriptiva namn som gör att granskaren av koden förstår vad det hänvisar till.
- Syntax rättas i enlighet med angivna standarder för varje språk.

2.1.2 Python

Ramverk

- Django 2.0.4 [1]

Standarder och riktlinjer

- PEP8 [2]

Struktur

- Endast enkla citationstecken (') ska användas.

Filnamn

- Samtliga filer ska ha filändelsen `*.py`.
- Om ett filnamn består av flera ord så ska dessa separeras med understreck.

Funktioner och klasser

- Om en funktion eller en klass består av flera ord så ska dessa separeras med understreck.
- Namngivning efter funktion, det vill säga efter vad funktionen eller klassen gör.

```
# Fel
def my_function(value1, value2):
     return value1 - value2

# Rätt
def subtract_values(value1, value2):
     return value1 - value2
```

Variabler

- Namngivning efter syfte, det vill säga vad de representerar.
- Om en variabel består av flera ord så ska dessa separeras med understreck.

Kommentarer

- Alla funktioner och klasser ska ha *docstrings* i enlighet med konventionen PEP 257, det vill säga åtminstone en enradskommentar och eventuellt en docstring på flera rader [2].
    - En enradskommentar i en docstring skrivs på samma rad som omgivande citationstecken.
    - En flerradskommentar i en docstring innehåller en inline-kommentar, följt av en blankrad och sedan en mer ingående beskrivning av funktionen eller klassen. Kommentaren avslutas med tre citationstecken i en ny rad.

```
def function(a, b):
    '''Do X and return a list.'''

def complex(real=0.0, imag=0.0):
    '''Form a complex number.

    Keyword arguments:
    real -- the real part (default 0.0)
    imag -- the imaginary part (default 0.0)
    '''
    if imag == 0.0 and real == 0.0:
        return complex_zero
```

- Övriga kommentar ska skrivas med hashtags, och indenteras till samma nivå som koden som hänvisas.

```
# This is how to write one or several
# lines of comments in Python.
answer = 42
```

- Kommentarer som skrivs på samma rad som kod bör undvikas, med undantag för om det på ett kortfattat och nödvändigt sätt beskriver syftet till en enda kodrad.

Kommentaren separeras då från koden med en tabb, motsvarande fyra mellanrum. PEP8 anger egentligen minst två mellanrum [2], men eftersom avstånd i projektets kod tenderar att vara en tabb lång, motsvarande fyra mellanrum, så kommer det tillämpas även här.

```
x = x + 1    # Compensate for border
```

2.1.3 SQL

Databashanterare

- PostgreSQL [3]

Standarder och riktlinjer

- PostgreSQL:s officiella dokumentation [4]
- SQL Style Guide av Simon Holywell [5]
    - Riktlinjer kring struktur, filnamn, tabeller, kolumner, kommentarer och övrig utformning listas och redogörs i källan.

Filnamn

- Samtliga filer ska ha filändelsen `*.sql`.

2.1.4 HTML

Ramverk

Eftersom ramverket Django kommer användas för projektets Python-kod, tillämpas även det mallspråk som ingår [1]. Django Template Language består av en kombination av HTML och Python, där Python-funktioner skrivs inom måsvingar {...} och procenttecken (%) medan variabler står inom dubbla måsvingar {{...}}.

```
<ul>
{% for i in users %}
      <li>{{ first_name }}</li>
      <li>{{ last_name }}</li>
{% endfor %}
</ul>
```

Endast ett dokument – standardmallen – innehåller fullständig HTML-kod, vilket åtminstone bör innehålla följande:

- `<!DOCTYPE html>`
- `<html lang=”sv”>...</html>`
- `<head>...</head>`
    - `<title></title>`
    - `<meta charset=”utf-8”>`
    - `<meta name="viewport" content="width=device-width, initial-scale=1.0">`
- `<body>...</body>`
- `<div id=”app”>...</div>`
- `<header>...</header>` med komplett innehåll
- `<aside>...</aside>` med komplett innehåll
- `<main>...</main>`
    - `{% block main %}{% endblock %}`

Det sistnämnda blocket är platshållare för innehåll som kan komma att fyllas i ett malldokument, som därefter “klistras in” i blockets definierade plats i standardmallen. Samtliga block ska namnges i starttaggen.

```
# Fel
{% block %}{% endblock %}
```

```
# Rätt
{% block title %}{% endblock
%}
```

Detta tillvägagångssätt är i enlighet med mjukvaruprincipen Don’t Repeat Yourself (DRY), som syftar till att effektivisera kodskrivandet och att dela systemet i bitar för att minska

komplexiteten hos helheten [6]. Nedan är ett exempel på standardmallen följt av ett malldokument där det först deklareras att filen är en förlängning av `index.html`.

```
# layout.html
<!DOCTYPE html>
<html lang=”sv”>
    <head>
        <meta charset=”UTF-8”>
        <title>My Website</title>
    </head>
    <body>
        <main>
            {% block main %}
        </main>
    </body>
</html>

# index.html
{% extends “layout.html” %}
    {% block main %}
        <p>Welcome to my website!</p>
    {% endblock %}
```

Standarder och riktlinjer

- HTML 5.2 av World Wide Web Consortium (*W3C*) [7]
- Web Content Accessibility Guidelines (*WCAG*) v 2.1 av World Wide Web Consortium (*W3C*) [8]

Struktur

- Dubbla citationstecken (“) ska användas.
- Alla HTML-dokument ska inledas med `<!DOCTYPE html>`.
    - Detta innebär att endast det standardiserande HTML-dokumentet behöver ha detta element, eftersom resterande mallar hänvisar tillbaka till det.
- Länken till CSS-filen ska placeras i `<head>`-sektionen och använda sig en inbyggd funktionalitet i Django som hänvisar till statiska filer.

```
        <link rel="stylesheet" href="{% static 'css/main.css'
%}">
```

- Indentering sker för varje element som ligger inom ett annat, vilket inkluderar `<head>` och `<body>` eftersom dessa ligger inom `<html>`.

Filnamn

- Samtliga filer ska sluta med filändelsen `*.html`.
- Om ett filnamn består av flera ord så ska dessa separeras med bindestreck.

Element

- Så många semantiska element som möjligt bör användas, istället för mer generella och intetsägande sådana (*icke-semantiska*).

Semantiska element		Icke-semantiska element
<nav>	<header>	
<main>	<aside>	
<footer>	<figure>	
<section>	<article>	

- Alla HTML-element ska vara stängda.

```
# Fel
<strong>A phrase tag defining important text.

# Rätt
<strong>A phrase tag defining important text.</strong>
```

Attribut

- Klasser och ID ska skrivas i gemener.
- Om en klass eller ett ID innehåller flera ord så ska dessa separeras med bindestreck.

```
# Fel
```

```
<p class="CARDTEXT">...</p>

# Rätt
<p class="card-text">...</p>
```

- En klass eller ett ID ska ges till de element som, på grund av sin semantiska innebörd, används regelbundet, och därför behöver särskiljas. Om föräldern till ett sådant element har ett givet attribut är det emellertid inte nödvändigt att ge elementet ett eget attribut. Detta för att det lätt kan bli ett överflöd av klasser och ID.

```
# Fel
<nav class="primary-menu">
     <ul class="main-menu">
          <li class="list-item">...</li>
          <li class="list-item">...</li>
     </ul>
</nav>

# Rätt
<nav class="primary-menu">
     <ul>
          <li>...</li>
          <li>...</li>
     </ul>
</nav>
```

- Ett ID ges till de element som är ämnade att förekomma i endast en upplaga per HTML-dokument. Dessutom attribueras ett ID till de element som behöver vara åtkomliga i JavaScript.

```
<div id="wrapper">
     …
</div>
```

- En klass ges till de element som förekommer eller kan förekomma i mer än en upplaga per HTML-dokument.

```
<a href="#" class="external-link">...</a>
```

Kommentarer

Kommentarer kommer i projektets HTML-dokument att skrivas på tre olika sätt:

1. **För att markera var ett icke-semantiskt element tar slut.** Sluttaggen för generella element säger inte mycket om vilken typ av innehåll som precis avslutades. Därför implementeras kommentarer inom `<!--  /` och `-->` för att förtydliga koden och förbättra dokumentets läsbarhet. Är det en klass som avslutas markeras detta genom en punkt (.) efter snedstrecket. Är det istället ett ID det gäller, så är det en hashtag (#) som följer snedstrecket istället.

```
<section class=”profile”>
      …
</section> <!-- /.section -->
```

2. **För att förtydliga när koden inte är självförklarande.** Icke-semantiska element kan – trots deskriptiva attribut – behöva ytterligare förklaring.

```
<!-- Comments like these can be written on one
or several lines -->
```

3. **För att annotera arbetsanteckningar för utvecklarna.** Dessa typer av kommentarer är till för anteckningar kring framtida eller önskade tillägg i koden, eventuella frågetecken eller påminnelser till utvecklarna. Kommentarerna skrivs i enlighet med kommentarsfunktionen i Django Template Language genom att omges av måsvingar, hashtags och ett mellanrum [1].

```
{# Working notes #}
```

2.1.5 CSS

Preprocessor

Vad gäller styling så kommer webbapplikationen Paca utvecklas med hjälp av *preprocessorn* Sass [9], som används för att presentera strukturerat innehåll och ger utökad funktionalitet i jämförelse med CSS. Detta inkluderar funktioner såsom att använda variabler, nästlade selektorer och matematiska uttryck.

Sass fördelar arbetet med stilmallen i flera mappar och filer, vilket erbjuder en översiktlig och logisk struktur. Av den anledningen ligger inte samma krav på att slutprodukten (CSS-filen) ska struktureras upp med innehållsförteckning och diverse avdelare, eftersom det snarare är samlingen av `*.scss`-filer – och inte den ensamstående `*.css`-filen – som läsare granskar och utvecklare arbetar i.

Standarder och riktlinjer

- Standarder från W3C
    - CSS Snapshot 2017 [10]
    - Selectors Level 3 [11]
    - CSS Fonts Module Level 3 [12]
    - CSS Backgrounds and Borders Module Level 3 [13]
    - CSS Color Module Level 3 [14]
    - WCAG 2.1 [8]
- Sass:s officiella dokumentation [9]

Struktur

- Varje CSS-regel ska separeras med en blankrad.
- Om en regel innehåller flera selektorer, ska dessa skrivas på varsin rad.

- Alla *deklarationer* i deklarationsblocket ska indenteras med en tabb, motsvarande fyra mellanslag, åt höger.

```
# Fel
i, cite, figcaption { font-style: italic; }

i, cite, figcaption {
      font-style: italic;
}

# Rätt
i,
cite,
figcaption {
      font-style: italic;
}
```

Filnamn

- Samtliga filer ska börja med eller utan ett understreck.
    - Ett understreck i filnamnets första position indikerar att filen är en komponent som ska importeras till ett annat dokument.
    - Frånvaron av ett understreck innebär att Sass-dokumentet är ett komplett dokument (i regel bestående av en rad påståenden som importerar individuella komponenter).
- Samtliga filer ska sluta med filändelsen `*.scss`. Efter konvertering till CSS kombineras alla dokument till en enhetlig `*.css`-fil.

Selektorer

- En kombination av nästlade selektorer och klasser eller ID bör användas.
    - För många klasser och ID gör att dokumenten blir intetsägande, eftersom överblicken av hierarkier går förlorad.

```
# Fel
.primary-nav {}
.menu {...}
```

```
# Rätt
.primary-nav {}
.menu {
     ...
```

```
.menu-list-item {...}
.menu-a {...}
.sub-menu {...}
.sub-menu-list-item {...}
.sub-menu-a {...}
```

```
    li {
        ...
        a {}
    }
}
.sub-menu {
    li {
        ...
        a {}
    }
}
```

- För djupt nästlade selektorer kan dock skapa problem vid omorganisering i koden, eller åtminstone ge en bristande överblick om selektorerna inte är meningsbärande. Därför bör nästlade regler aldrig vara djupare än fyra nivåer [15].

Variabler

Här kommer det eventuellt skrivas om variabler i en senare sprint, när projektgruppen tagit beslut kring i vilken utsträckning variabler kommer att användas i Sass [9].

Kommentarer

I Sass kan kommentarer skrivas på två sätt [9]:

1. **Enradskommentarer** skrivs efter två bindestreck, minst en tabb till höger, på samma rad som den hänvisade koden. De bibehålls inte vid konvertering till CSS, och är därför endast lämpade för arbetsanteckningar av och för utvecklarna.
2. **Flerradskommentarer** skrivs inom ett snedstreck och en asterix. De bibehålls vid konvertering till CSS, och är därför lämpade för förtydligande kommentarer.

2.1.6 JavaScript

Ramverk

- jQuery [16]

Plugins

- FullCalendar.js [17]

Standarder och riktlinjer

- ECMAScript [18]
- JavaScript Standard Style av Feross Aboukhadijeh [19]
    - Riktlinjer kring struktur, regler, namngivning, kommentarer och övrig utformning listas och redogörs i källan.

Struktur

- Samtliga jQuery-funktioner ska inledas med ett dollartecken ($).

Filnamn

- Samtliga filer ska sluta med filändelsen `*.js`.

2.2 Granskningsmöten

Den metodologiska utgångspunkt som projektets granskningsmetod baseras på beskrivs av Frank Tsui, Orlando Karam och Barbara Bernal i “Essentials of Software Engineering” [20]. Granskningsprocessen delas här in i sex steg:

1. Planering
    - Granskningsgruppen, bestående av tre till fyra personer [21], en moderator och en läsare bestäms. Läsaren får inte vara den primära utvecklaren av koden.
    - Moderatorn ser till att koden är exekverbar, och därmed går att testas.
    - Moderatorn delar ut granskningsmaterialet till alla medlemmar i granskningsgruppen i god tid.
2. Översikt
    - Granskningsmaterialet ska presenteras översiktligt såvida inte alla medlemmar i granskningsgruppen är bekanta med koden.
3. Förberedelse
    - Varje granskare ska, med hjälp av checklistan för kodgranskning, studera granskningsmaterialet inför granskningsmötet.
4. Granskning
    - Granskningsmötet pågår mellan en och två timmar.
    - Den utvalda läsaren presenterar granskningsmaterialet.
    - Granskarna granskar materialet tillsammans med hjälp av checklistan för kodgranskning.
    - Brister och fel påpekas och rapporteras i checklistan för kodgranskning. Diskussioner kring anledningen bakom en defekt eller tillvägagångssätt för åtgärder hör inte hemma under ett granskningsmöte.
    - Granskningsmaterialet accepteras antingen som det är, eller efter resterande steg i granskningsprocessen.
5. Omarbete
    - En eller flera ansvariga utvecklare korrigerar eventuella brister och fel i det granskade materialet.
6. Uppföljning
    - Korrigerat material kontrolleras av moderator, eller granskas igen.

3. Process

3.1 Kontinuerlig kodgranskning

För att säkerställa kvalitet, utan att kräva alltför mycket tid från gruppens tidsbudget, kommer kodfiler kontinuerligt att granskas med hjälp av webbaserade valideringsverktyg. Dessa kontrollerar enbart koden i enlighet med den standard som verktyget rättar sig efter, och resulterar alltså inte i en fullständig granskning.

Validering av Python-kod kommer ske genom att klistra in kod eller ladda upp *.py-dokument på PEP8 Online [22]. Både HTML- och kompilerad CSS-kod kommer verifieras med hjälp av de två valideringsverktyg som W3C tillhandahåller: Markup Validation Service [23] respektive CSS Validation Service [24].

3.2 Granskningsmöte

Mellan tre och fyra medlemmar av projektgruppen ska i slutet av andra sprinten, samt fler gånger under resterande projekttid, mötas under högst två timmar för att genomgå en *peer-review* där produktens kod granskas för brister och defekter [21].

Inför mötet utses en moderator, som dessutom tar sig an rollen som läsare, och en sekreterare. Moderatorn går igenom checklistan för kodgranskning, medan sekreteraren skriver ner eventuella brister och fel i ett protokoll som följer den mall som presenteras i 3.2.2.

Checklistan för kodgranskning är baserad på den metod som presenterades i slutet av föregående avsnitt (2.2). Metoden förbjuder diskussioner kring eventuella åtgärder eftersom mötet inte får överskrida två timmar, men under detta projekt kommer det snarare

avrådas eftersom en granskare kanske kan lösningen till ett problem – och således avlägsnar behovet av en ansvarig för åtgärder.

När ett problem skrivs i "Sammanfattning av defekter" ska det även ges en allvarsgrad enligt nedanstående lista. Allvarsgrad 1 kräver omedelbara åtgärder, medan allvarsgrad 2 inte är fullt lika prioriterad eftersom det inte förhindrar koden att köras över huvud taget. Allvarsgrad 3 gäller problem av enbart semantisk karaktär, såsom inkorrekt kommenterade funktioner, och bör därför åtgärdas efter högre prioriteringar för att koden ska vara så förståelig och tillgänglig som möjligt.

1. Åtgärd måste vidtas för att koden ska fungera över huvud taget
2. Åtgärd bör vidtas för att koden ska fungera helt som förväntat
3. Åtgärd kan skjutas upp eftersom det inte har en direkt inverkan på exekveringen av koden

3.2.1 Mall för protokoll

Varje mötesprotokoll ska innehålla en upplaga av "Metadata" och "Sammanfattning av defekter", samt en upplaga av "Checklista för kodgranskning" för varje kodfil som granskas.

Granskningsprotokoll <ID för protokollet>

Metadata	
Tid och datum	<hh:mm> <ååmmdd>
Plats	<T.ex. salsnummer>
Ansvarig utvecklare	<Namn>
Moderator	<Namn>
Sekreterare	<Namn>
Övriga granskare	<Namn>

Granskade filer	<Filnamn>

Checklista för kodgranskning:

Nr	Kategori	Fråga	JA	NEJ
1	Standarder och riktlinjer	Följer koden de standarder och riktlinjer som definierats för programspråket?		
2	Standarder och riktlinjer	Validerar koden i ett givet webbverktyg för programspråket?		
3	Struktur	Är koden lättläst?		
4	Struktur	Är koden korrekt indenterad?		
5	Namngivning	Är filnamnet lämpligt?		
6	Namngivning	Är formatet på namngivningen lämplig i koden?		
7	Namngivning	Är den mening som tillskrivs vid namngivning korrekt?*		
8	Kommentarer	Är kommentarer skrivna på korrekt sätt?		
9	Kommentarer	Finns det kommentarer på de ställen som förväntas ha kommentarer?		

* T.ex. ska funktioner i Python namnges efter funktion, medan variabler ska namnges efter vad deras värde syftar till.

Sammanfattning av defekter

Nr	Filnamn	Rad	Typ av problem	Allvars-grad	Ansvarig för åtgärd
<#>	filnamn.filändelse	#	<Beskrivning av problem.>	<1-3>	<Namn>

3.2.3 Granskningsprotokoll

Nedan presenteras rubriker samt en referens till den bilaga där hela protokollet står.

Granskningsprotokoll <Id för protokollet>

Se bilaga #.

Granskningsprotokoll <Id för protokollet>

Se bilaga #.