


Koncept

- Úvod
- Proces
 - Udalosť
- Aplikácia
- Namespace
- Petriflow
 - Proces
 - Udalosti
 - Platnosť
 - Inštancia
 - Udalosti
 - Miesto
 - Statické miesto
 - Úloha
 - Statická úloha
 - Formulár
 - Layout
 - Trigger
 - Udalosti
 - Hrana
 - Regular
 - Inhibitor
 - Reset
 - Read
 - Variabilné hrany
 - Rola (User ref)
 - Statická Rola
 - Dátová premenná
 - Statická dátová premenná
 - Počiatočná hodnota
 - Validácia
 - Komponent
 - Udalosti
 - Šifrovanie
 - Dátové pole
 - Správanie
 - Štýl
 - Komponent
 - Udalosti
 - Akcia
 - Funkcia
 - Statická funkcia
 - Výraz
 - I18n
- Netgrif Application Engine
 - Deployment
 - Action API
 - Kontext
 - Komponenty
 - Text komponenty
 - Number komponenty
 - Enumeration komponenty
 - Multichoice komponenty
 - File komponenty
 - Button komponenty
 - Frontendové akcie
 - Presmerovanie
 - Otvorenie dialógu
 - Prompt
 - Confirm
 - Alert
 - Task
 - Otvorenie/zatvorenie side menu
 - Otvorenie/zatvorenie expansion panelu
 - Otvorenie/zatvorenie tabu
 - Event, EventOutcome
 - Container - Component
 - Filtre
 - Používateľ
 - Grupy

- Expression init a validacie
- Rule Engine
- SVG komponent
- Event sourcing & logging
- Websockety
- Single Sign-On
- Case Indexing
- Dashboard
- Properties
- Mail Service
 - Mail configuration
 - Mail draft
 - Action
- I18n
- Marketplace
- Academy / Developer Portal
 - Quick start
 - Guides/cookbook
 - Referencna prirucka
 - Technicka dokumentacia
 - Kurzy Petriflow / Netgrif

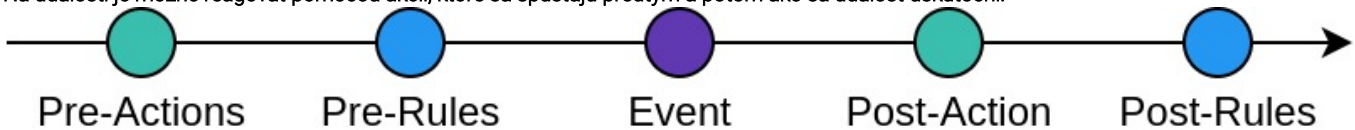
Úvod

 Definícia základných pojmov: proces, instancia, udalosť, aplikácia ...

Proces

Udalosť

Na udalosti je možné reagovať pomocou akcií, ktoré sa spúšťajú predtým a potom ako sa udalosť uskutoční.



Aplikácia

Súbor spolu súvisiacich procesov, ktoré spoločne riešia komplexný business problém

Namespace

Logický súbor spolu súvisiacich aplikácií. Proces musí mať unikátne URL v rámci namespaceu v tvare `namespace:aplikacia:proces`.

Petriflow

Proces

Proces musí mať v rámci jednej aplikácie unikátne `id`. Názov procesu `title` je textový reťazec typu `i18n`. Proces môže definovať ikonu `icon`, ktorá sa aplikuje aj na jednotlivé inštalácie daného procesu.



```

<process>

<id>process_id<
/id>
  <title ="
process_title"
>Demo Process<
/title>
  <initials>DP<
/initials>
  <icon>home<
/icon>
  ...
</process>

```

Udalosti

Na procese sú definované udalosti:

- nasadenie procesu `deploy`
- zmazanie procesu `delete`

```

<process>
...

<processEvents>
  <event
type="
deploy">
...
  </event>
<
/processEvents>
...
</process>

```

Platnosť

Proces má nastaviteľné obdobie odkedy dokedy je platný. Štandardne sa po nasadení procesu nastaví aktuálny dátum ako začiatok platnosti a koniec platnosti sa nenastaví. Ak sa jedná o novú verziu procesu tak sa predošlej nastaví koniec platnosti rovnaký ako začiatok platnosti novej verzie. Začiatok platnosti je možné nastaviť pri importovaní procesu. Nové inštalácie sa vytvára vždy len z aktuálne platnej verzie procesu.


Inštalácia

Inštalácia je jedným spustením daného procesu. Všetky nestatické objekty sa ukladajú v rámci jednej inštalácie.

Udalosti

Na procese sú definované udalosti:

- vytvorenie novej inštalácie `create`,

 TODO: schema

- zobrazenie inštancie `view`,
- zmazanie inštancie `delete`.

```
<process>
  ...

  <caseEvents>
    <event
type="
create">
      ...
    </event>
    <event
type="
delete">
      ...
    </event>
  <
/caseEvents>
  ...
</process>
```

Miesto

Miesto musí mať v rámci procesu unikátne `id`. Pre grafickú vizualizáciu definuje miesto svoje súradnice ako `x`, `y` a názov ako `label`. Miesto môže obsahovať nezáporný celočíselný počet značiek (tokenov). Počiatočná hodnota značkovania je definovaná nezápornou celočíselnou hodnotou `tokens`.

```
<process>
  ...
  <place>
    <id>p1</id>
    <x>100</x>
    <y>200</y>

    <label>Start<
  /label>
    <tokens>1<
  /tokens>
  </place>
  ...
</process>
```

```

<xs:element
name="place">
  <xs:
complexType>
  <xs:
sequence>
  <xs:
element name="
id" type="xs:
string"/>
  <xs:
element name="
x" type="
nonNegativeInte
ger"/>
  <xs:
element name="
y" type="
nonNegativeInte
ger"/>
  <xs:
element name="
label" type="
i18nStringType"
minOccurs="0"/>
  <xs:
element name="
tokens" type="
nonNegativeInte
ger"
minOccurs="0"
default="0"/>
  <xs:
element name="
static" type="
xs:boolean"
minOccurs="0"
default="false"
/>
  </xs:
sequence>
</xs:
complexType>
</xs:element>

```

Statické miesto je miesto, ktorého značkovanie je zdieľané naprieč všetkými inštanciami daného procesu. Statické miesto je definované pravdivou hodnotou `isStatic`.

```
<place>
  <id>p1<
/ id>
  <x>100<
/ x>
  <y>200<
/ y>

<label>Start
</label>

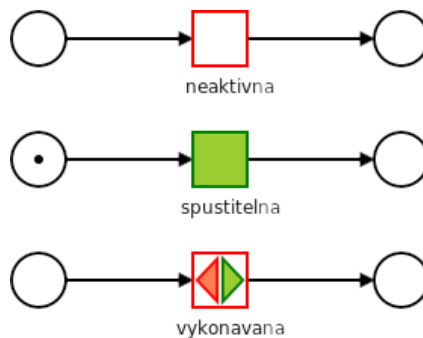
<tokens>1<
/ tokens>

<static>true
</static>
</place>
```

Úloha

Petriflow definuje úlohy ako rozšírenie prechodov v Petriho sieti. Úloha existuje po celý život inštancie. Pri vytvorení novej inštancie sa vytvoria všetky úlohy a podľa značkovania sa im nastaví stav. Stavy sú:

- neaktívna - ak je prechod v Petriho sieti nespustiteľný,
- spustiteľná - ak je prechod v Petriho sieti spustiteľný a úloha nie je vykonávaná používateľom,
- vykonávaná - ak si úlohu priradil používateľ a vykonáva ju.



```
<transition>
  <id>t1</id>
  <x>100</x>
  <y>200</y>

<label>Start<
/ label>
  <tokens>1<
/ tokens>

<static>true<
/ static>
</transition>
```

Statická úloha

Formulár

K úlohe je možné naviazať formulár, v ktorom sa budú zobrazovať jednotlivé dátové polia alebo vnorené formuláre. Každý formulár môže špecifikovať vlastný layout do ktorého sa dátové polia rozložia. Ak nie je layout špecifikovaný použije sa layout nadradeného formuláru.

Layout

Jediným podporovaným layoutom je grid layout. Konfiguráciou je možné nastaviť:

- rozmery layoutu: fit to screen, vertical scroll, horizontal scroll, fixed, vertical fixed, horizontal fixed
 - výšku riadku
 - šírku riadku
 - počet stĺpcov

⚠ TODO

⚠ TODO

- mobile breakpoint
- zarovňavanie fieldov: žiadne, hore, hore+vľavo, hore+vpravo, vľavo, vľavo+hore, vpravo, vpravo+hore
 - ak hore
 - vynechávajúce celých prázdných riadkov: áno, nie

Trigger


Úlohe je možné nastaviť automatizované spustenie pomocou triggerov. Petriflow definuje dva typy triggerov: *auto* a *time*. Auto trigger slúži na okamžité vykonanie úlohy akonáhle sa dostane do stavu spustiteľná. Time trigger zabezpečí spustenie úlohy v špecifikovanom čase ak je vtedy úloha spustiteľná. Time trigger je možné nakonfigurovať dvomi spôsobmi ako *exact* a *delay*. Ak sa nastaví time trigger *exact* tak hodnotou je presný dátum a čas, kedy sa má úloha spustiť. Ak sa nastaví *delay* time trigger tak hodnotou je oneskorenie s akým sa má úloha spustiť od momentu kedy je spustiteľná.

Úlohe je možné nastaviť aj čiastočné trigger na niektoré udalosti. Petriflow definuje trigger na *assign* a *finish*. Na *assign* trigger je možné nakonfigurovať či sa má spustiť automaticky po udalosti *open* danej úlohy. Finish trigger je možné nakonfigurovať tak aby sa úloha po udalosti *assign* automaticky dokončila ak neobsahuje žiadny formulár.

Udalosti

Na úlohe sú definované udalosti:

- zobrazenie úlohy *view*,
- začiatok vykonávania *assign*,
- dokončenie úlohy *finish*,
- zrušenie vykonávania *cancel*,
- otvorenie úlohy *open*,
- zatvorenie úlohy *close*.

 TODO


```
<transition>
  <id>task</id>
  ...
  <trigger
type="auto" />
  ...
</transition>
<transition>
  <id>task</id>
  ...
  <trigger
type="auto" />
  ...
</transition>
```

```
<transition>
  <id>task<
/id>
  ...
  <event
type="
assign">

  <id>task_ass
ign</id>
  ...
  </event>
  ...
<
/transition>
```

Hrana

Petriflow definuje 4 základné typy hrán: *regular*, *inhibitor*, *reset*, *read*.

 poradie vyhodnocovania

Regular

⚠️ TODO

⚠️ TODO: obr

```
<process>
  ...
  <arc>
    <id>a1</id>

    <type>regular<
  /type>

  <sourceId>p1<
  /sourceId>

  <destinationId>
  t1<
  /destinationId>
    </arc>
  ...
</process>
```

Inhibitor

Inhibitor hrana má opačnú podmienku spustiteľnosti ako obyčajná hrana, prechod je spustiteľný ak je vo vstupnom mieste menej tokenov ako je násobnosť inhibitor hrany. Pri spustení prechodu sa nekonzumujú žiadne tokeny.

⚠️ TODO: obr

```
<process>
  ...
  <arc>
    <id>a1</id>

    <type>inhibitor
  </type>

  <sourceId>p1<
  /sourceId>

  <destinationId>
  t1<
  /destinationId>
    </arc>
  ...
</process>
```

Reset

Reset hrana nemá žiadnu podmienku spustiteľnosti. Pri spustení prechodu sa konzumujú všetky tokeny.

⚠️ TODO: obr

```
<process>
  ...
  <arc>
    <id>a1</id>
```



```

<type>reset<
/type>

<sourceId>p1<
/sourceId>

<destinationId>
t1<
/destinationId>
</arc>
...
</process>

```

Read

Read hrana má rovnakú podmienku spustiteľnosti ako obyčajná hrana. Pri spustení prechodu sa nekonzumujú žiadne tokeny.



```

<process>
...
<arc>
  <id>a1</id>
  <type>read<
/type>

<sourceId>p1<
/sourceId>

<destinationId>
t1<
/destinationId>
</arc>
...
</process>

```

Variabilné hrany

Každá hrana môže byť zároveň aj variabilnou hranou, pričom násobnosť hrany je určená buď počtom tokenov v referencovanom mieste alebo hodnotou celočíselného dátového poľa v inštancii. Pri deployovaní sa ako prvé pozerá na miesta v danom procese a hľadá sa miesto s daným id. Ak sa nenájde vhodné miesto začnú sa prehľadávať dátové premenné.

```

<process>
...
<arc>
  <id>a1<
/id>

<type>regular</type>

<reference>number_field_

```

```
id<
/reference>
...
</arc>
...
</process>
```

Rola (User ref)

⚠️ TODO

Statická Rola

⚠️ TODO

Dátová premenná

⚠️ TODO

Statická dátová premenná

⚠️ TODO

Počiatočná hodnota

Dátovej premennej sa dá nastaviť počiatočná hodnota pomocou `init`. Hodnotou môže byť konštanta alebo výraz. Ak je hodnotou výraz musí byť nastavený atribút `dynamic` na hodnotu `true`. Počiatočná hodnota sa nastavuje pri vytváraní inštalácie.

⚠️ TODO staticke?

```
<data type="number">
  <id>number_0</id>
  ...
  <init>10</init>
</data>
<data type="number">
  <id>number_1</id>
  ...
  <init dynamic="true"
>${number_0 * 100}</init>
</data>
```

Validácia

⚠️ TODO

Komponent

Dátovú premennú je možné vo formulári zobrazit vo forme rôznych grafických komponentov. Komponent nastavený na dátovej premennej sa zobrazí v každom formulári, kde nie je použitý iný komponent. Ďalšie grafické úpravy je možné definovať v konfigurácii komponentu `properties`, kde je možné nastavovať rôzne CSS atribúty.

```
<data type="
enumeration"
```

```

>

<id>icon_enu
m</id>
  <title><
  /title>
  <options>
    <option
key="1"
>Krádež<
  /option>
    <option
key="2"
>Poškodenie
Zariadenia<
  /option>
    <option
key="3"
>Rozbitie
skla<
  /option>
    <option
key="4"
>Vandalizmus
  </option>
    <option
key="5"
>Živelná
škoda<
  /option>
  </options>

<component>

<name>icon<
  /name>

<properties>

<option_icons>

  <icon key="
1">home<
  /icon>

  <icon key="
2">nature<
  /icon>

```

```

<icon key="
3">home<
/icon>

<icon key="
4">nature<
/icon>

<icon key="
5">home<
/icon>
      <
      /option_icon
s>
      <
      /properties>
      <
      /component>
</data>

```

Udalosti

Na dátovej premennej sú definované udalosti:

- načítanie hodnoty `get`,
- uloženie novej hodnoty `set`.

Tieto udalosti sa volajú pri volaní danej udalosti na ľubovoľnom formulári.

```

<data type="
text">
  <id>text<
  /id>

  <title>Text<
  /title>
  ...
  <event
  type="get">

  <id>get_even
  t_id</id>
  ...
  </event>
  <event
  type="set">

  <id>set_even
  t_id</id>
  ...
  </event>
</data>

```

Šifrovanie

Na úrovni dátovej premennej je možné definovať šifrovanie na úrovni databázy pomocou `encryption`. Atribútom `algorithm` je navyše možné definovať aj šifrovací algoritmus. Defaultná hodnota je `false` a default algoritmus je `PBEWITHSHA256AND256BITAES-CBC-BC`.

```
<data type="text">
  <id>text</id>
  <title>Text</title>
  <encryption algorithm=" "
>true</encryption>
</data>
```

Dátové pole

⚠ TODO

Správanie

Na dátové pole je možné definovať jeho správanie v danom formulári. Definované správania sú:

- `forbidden` - dátové pole sa vo formulári nezobrazuje a nevykonáva sa `get` udalosť dátového poľa,
- `hidden` - dátové pole sa vo formulári nezobrazuje, vykonáva sa `get` udalosť dátového poľa,
- `visible` - dátové pole je vo formulári zobrazené,
- `editable` - povoľuje zmenu hodnoty dátového poľa, umožňuje volať `set` udalosť dátového poľa,
- `required` - dátové pole musí obsahovať hodnotu aby sa úloha mohla dokončiť,
- `optional` - dátové pole nemusí obsahovať hodnotu aby sa úloha mohla dokončiť.

	for bid den	hid den	visi ble	edi tab le	req uir ed	opt ion al
forbi dden	✗	✗	✗	✗	✗	✗
hidd en	✗	✗	✗	✗	✓	✓
visib le	✗	✗	✗	✗	✓	✓
edita ble	✗	✗	✗	✗	✓	✓
requ ired	✗	✓	✓	✓	✗	✗
opti onal	✗	✓	✓	✓	✗	✗

⚠ TODO: default?

Štýl

Komponent

Rovnako ako na dátovej premennej aj na dátovom poli je možné definovať grafický komponent. Komponent definovaný na dátovom poli má vždy prednosť pred komponentom definovanom na dátovej premennej.

```
<data type="
enumeration"
>

<id>icon_enu
m</id>
  <title><
/ttitle>
  <options>
    <option
key="1"
>Krádež<
```

```

/option>
    <option
key="2"
>Poškodenie
Zariadenia<
/option>
    <option
key="3"
>Rozbitie
skla<
/option>
    <option
key="4"
>Vandalizmus
</option>
    <option
key="5"
>Živelná
škoda<
/option>
    </options>
</data>
...
<transition>
    ...
    <dataRef>

<id>icon_enu
m</id>
    <logic>

<behavior>ed
itable<
/behavior>
    </logic>

<component>

<name>icon<
/name>

<properties>

<option_icons>

<icon key="
1">home<
/icon>

```

```

<icon key="
2">nature<
/icon>

<icon key="
3">home<
/icon>

<icon key="
4">nature<
/icon>

<icon key="
5">home<
/icon>
      <
    /option_icon
s>
      <
    /properties>
      <
    /component>
      </dataRef>
      ...
    <
  /transition>

```

Udalosti

Na dátovom poli sú definované udalosti:

- načítanie hodnoty `get`,
- uloženie novej hodnoty `set`.

Tieto udalosti sa volajú iba pri načítaní a uložení hodnoty na danom formulári.

```


<dataRef>
  <id>text<
  /id>
  ...
  <event
type="get">

  <id>get_even
t_id</id>
  ...
  </event>
  <event
type="set">

  <id>set_even
t_id</id>
  ...
  </event>
</dataRef>


```

Akcia

 TODO

Funkcia

V procese je možné definovať funkcie, ktoré sa následne dajú použiť v akciách.

 TODO

Statická funkcia

```
<process>
  ...
  <function
static=true>
    double
    calculateMortga
ge(double
loan, double
period) {
    return
    (loan + loan.
value * 0.02 *
period.value)/
(period.value
* 12)
    }
  </function>
  ...
  <action>
    loan: f.
loan_amount,
    period:
f.period,
    monthly:
f.
monthly_payment
;
    change
monthly value {

    calculateMortga
ge(loan.value,
period.value)
    }
  </action>
  ...
</process>
```

Výraz



TODO

i18n

Petriflow definuje textové reťazce, ktoré môžu byť preložené do rôznych jazykov v závislosti od jazykového nastavenia používateľa. Takýto reťazec má atribút `name`, ktorý slúži na identifikáciu reťazca v danom procese. Proces definuje preklady pre jeden jazyk v objekte `i18n`. Tento objekt má atribút `locale`, ktorý definuje jazyk prekladu, a obsahuje zoznam prekladov, ktoré predstavujú dvojicu `name : string` kde `name` je atribút textového reťazca na ktorý sa daný preklad viaže a `string` je preklad reťazca do daného jazyka.

```
<data type="text">
```

```
<id>first_name</id>
```

```
<title name="first_name">Name</title>
```

```
</data>
```

```
...
```

```
<i18n locale="sk">
```

```
<i18nString name="first_name">
```

```
>Meno</i18nString>
```

Netgrif Application Engine

TODO

Deployment

Action API

Kontext

Komponenty

Zoznam aktuálne podporovaných komponentov v NAE:

Text komponenty

`password`, `textarea`, `richtextarea`, `htmltextarea`

Number komponenty

`currency`

Enumeration komponenty

`list`, `autocomplete_dynamic`, `autocomplete`, `stepper`, `icon`

Multichoice komponenty

list

File komponenty

preview

Button komponenty

raised, stroked, flat, icon, fab, minifab

Frontendové akcie

Možnosť z akcie zavolať funkciu na frontende. Frontendové akcie sa propagujú cez `EventOutcome` objekty.

Presmerovanie

Pomocou akcie je možné používateľa presmerovať na iné zobrazenie alebo na ľubovoľnú adresu. Syntax volania:

```
redirect(String uri)
```

```
<action>
redirect('all_cases
/60ae327faae72113f64490df
/60ae329d69101c75b0489950')
redirect('google.com')
</action>
```

Otvorenie dialógu

Definované sú nasledovné typy dialógového okna:

- prompt
- confirm
- alert
- task

Prompt

Prompt dialóg slúži na získanie jednoduchej textovej odpovede od používateľa. Dialóg obsahuje text správy, vstupné pole pre zadanie odpovede a dve tlačidlá pre potvrdenie odpovede a zrušenie dialógu. Dialógu je potrebné nastaviť referenciu na dátové pole, ktorému sa nastaví hodnota. Ak nie je zvolená iná počiatočná hodnota použije sa aktuálna hodnota dátového pola. Syntax volania:

```
Dialog.alert(String title, String content, Field
inputField, String inputLabel = inputField.title,
String inputValue = inputField.value, okLabel =
"OK", cancelLabel = "Cancel")
```

```
Dialog.alert(String title, String content, String
inputId, String inputLabel, String inputValue =
null, okLabel = "OK", cancelLabel = "Cancel")
```

Prompt dialógu je možné nastaviť nasledovné atribúty:

- title - nadpis dialógu
- content - text tela dialógu
- inputLabel - názov vstupného pola
- inputId - id dátového pola ktorému sa má nastaviť hodnota zo vstupného pola
- inputValue - počiatočná hodnota vstupného pola
- okLabel - názov ok tlačidla
- cancelLabel - názov cancel tlačidla

Confirm

Confirm dialóg slúži na získanie kladnej/zápornej odpovede od používateľa. Dialóg obsahuje text správy a dve tlačidlá pre kladnú a zápornú odpoveď. Syntax volania:

```
<action>
Dialog.confirm("Delete", "Are
```

```
Dialog.confirm(String title, String content,
okLabel = "OK", cancelLabel = "Cancel"
```

Confirm dialógu je možné nastaviť nasledovné atribúty:

- title - nadpis dialógu
- content - text tela dialógu
- okLabel - názov ok tlačidla
- cancelLabel - názov cancel tlačidla

Alert

Alert dialóg slúži na zobrazenie upozornenia pre používateľa. Dialóg obsahuje text správy a jediné tlačidlo, ktorým sa dialóg zavrie. Syntax volania:


```
Dialog.alert(String title, String content,
buttonLabel = "OK")
```

Alert dialógu je možné nastaviť nasledovné atribúty:

- title - nadpis dialógu
- content - text tela dialógu
- button label - názov tlačidla

Task

Task dialóg slúžia na zobrazenie formulára úlohy.

 TODO

```
Dialog.task(String transitionId, Case case =
useCase, String title = null)
```

```
Dialog.task(Task task, String title = task.title)
```

```
Dialog.task(Transition task, String title = task.
title)
```

```
you sure you want to delete
the process?" )
</action>
```

```
<action>
Dialog.alert("Warning",
"Unsaved changes will be
lost")
</action>
```


```
<action>
Dialog.alert("t5", "Register")
</action>
```

```
<action>
uloha: t.t5;
Dialog.alert(uloha)
</action>
```

Otvorenie/zatvorenie side menu

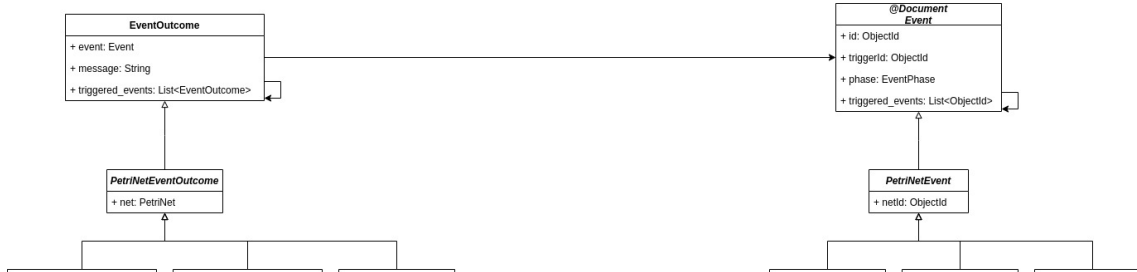
Otvorenie/zatvorenie expansion panelu

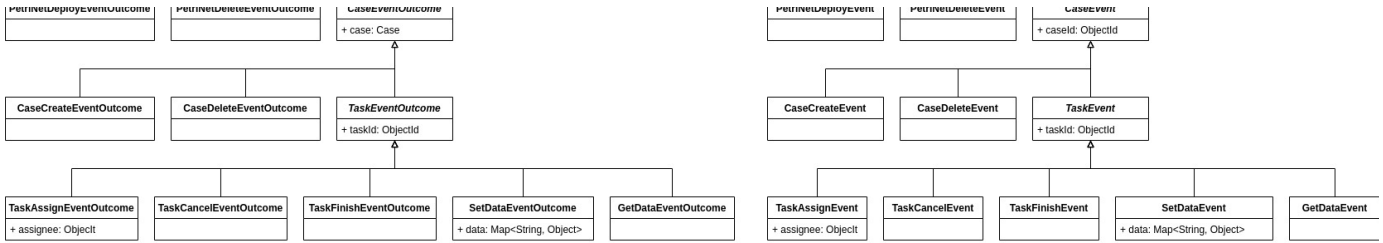
Otvorenie/zatvorenie tabu

 Doplnit vzor

Event, EventOutcome

Každá udalosť generuje objekt `Event`, ktorý je ukladaný do databázy ako inštancia procesu. Pre zobrazenie audit logu inštancie je postačujúce vyfiltrovať a zobrazíť tieto inštancie v štandardnom zobrazení. Každá udalosť si ukladá referenciu na entitu, ktorá ju vyvolala, fázu vykonávania a zoznam referencií udalostí, ktoré daná udalosť vyvolala. Na backende sa ako odpovede z volania udalostí používajú objekty `EventOutcome`, ktoré obaľujú `Event` a dopĺňajú referencie priamo na objekty.





Container - Component

Frontend pozostáva z komponentov, ktoré v sebe môžu obsahovať ďalšie komponenty. Frontend je možné vyskladať podobne ako formuláre v Builderi.

Filtre

⚠ TODO

Používateľ

⚠ TODO - anonym

Grupy

⚠ TODO - manazment, organizacna struktura, ...

Expression init a validacie

⚠ TODO

Rule Engine

⚠ TODO - import, export, syntax, kontext, ...

SVG komponent

⚠ TODO

Event sourcing & logging

⚠ TODO

Websockety

⚠ TODO

Single Sign-On

⚠ TODO



TODD - rozdelenie kolekcií podľa appky/procesu? problém s elasticom a 1000 indexami na jednu kolekciu, v cloude asi nepotrebné kvôli mikroservisom?

Dashboard

Properties

Mail Service

Baliček `com.netgrif.workflow.mail` obsahuje Java triedy pre prácu s emailami.

Mail configuration

Na odosielanie emailov sa používa trieda `org.springframework.mail.javamail.JavaMailSenderImpl`, ktorá je nakonfigurovaná podľa hodnôt v `application.properties`:

- `spring.mail.default-encoding`
- `spring.mail.host`
- `spring.mail.jndi-name.spring.mail.username`
- `spring.mail.jndi-name.spring.mail.password`
- `spring.mail.port`
- `spring.mail.properties.mail.debug`
- `spring.mail.properties.mail.smtp.debug`
- `spring.mail.properties.mail.smtp.auth`
- `spring.mail.properties.mail.smtp.starttls`
- `spring.mail.protocol`
- `spring.mail.test-connection`
- `spring.mail.smtp.starttls.enable`
- `spring.mail.smtp.starttls.required`

Mail draft

Trieda `MailDraft` predstavuje šablónu emailu. Trieda obsahuje nasledovné atribúty:

- `String from` - emailová adresa odosielateľa, môže byť odlišná než emailová adresa z ktorej sa email odosiela,
- `List<String> to` - zoznam emailových adries, ktorým má byť email poslaný,
- `List<String> cc` - zoznam emailových adries, ktorým má byť email poslaný ako kópia,
- `List<String> bcc` - zoznam emailových adries, ktorým má byť email poslaný ako skrytá kópia,
- `String subject` - predmet emailu,
- `String body` - telo emailu, môže byť jednoduchý text alebo HTML, telo môže obsahovať placeholder v tvare `${nazov}` a pri formátovaní sa placeholder nahradí textovou hodnotou objektu z mapy `model`,
- `boolean isHtml` - určuje či sa má telo emailu formátovať ako HTML, ak ale kolekcia `model` obsahuje aspoň jeden objekt tak sa automaticky email formátuje ako HTML automaticky,
- `Map<String, Object> model` - kolekcia objektov ktoré sú mapované podľa kľúča na placeholder v tele emailu,
- `Map<String, File> attachments` - kolekcia príloh emailu, kľúčom je názov súboru, ktorý sa bude zobrazovať v emaily a hodnotou je samotná príloha.

Action

```
void sendMail(MailDraft mailDraft)
```

I18n

Preklady sú uložené v zdrojovom modeli inštancie. REST API posiela v odpovedi vždy len konkrétny preklad daného `I18nString`-u v príslušnom Localised objekte.

- migrácie

Marketplace

Kategorizácia, hodnotenie, badge/award?

Academy / Developer Portal

Ak by prišiel nový developer tak ako prvé si prejde quickstart a rozbeha si NAE lokálne, následne začne prechádzať jednotlivé guidy a keď bude chcieť ísť viac do detailu tak referenčnú príručku. Ideálne by bolo mať jeden vzorový projekt na ktorom by bol vysvetlený koncept eTask + eForm a v priebehu pár hodín by cez to developer prišiel a vedel základy vývoja.

Quick start

Kratky a rychly navod ako stiahnuť a rozbehať NAE na lokálnom PC a spustiť jednoduchý "Hello world" projekt.

Guides/cookbook

Zameraná na vývoj konkrétneho projektu. Môže mať rôznu dĺžku - zameraný na celý projekt, alebo na riešenie nejakého frontendového problému, alebo na backendového problému, Ku každému guidu je verejne dostupný zdrojový kód daného projektu a dôležité časti a logika za nimi je popísaná v danom guide.

Referenčná príručka

Zameraná na jednotlivé moduly NAE (rule engine, mail, pdf, ...) a popisuje ako daný modul funguje z technického hľadiska, akú má konfiguráciu, ako sa dá používať a pod.

Technická dokumentácia

Vygenerované JavaDoc/GroovyDoc tried a ich funkcií engine backendu a frontendu.

Kurzy Petriflow / Netgrif

Kurzy vývoja aplikácií v Petriflow jazyku a Netgrif platforme dostupné na populárnych vzdelávacích platformách Udemy a Courser. Základný kurz Petriflowu je zadarmo, pozostáva zo 4 prednášok a obsahuje jedno praktické zadanie na precvičenie. Rozšírené kurzy Petriflow a Netgrif sú platené, na úspešné absolvovanie je potrebné vypracovať a úspešne obhájiť zadanie, následne účastník získava príslušný certifikát.