Development

Aanpassen huisstijl

De eerste wijziging die is doorgevoerd, is het uitbreiden van de huisstijl configuratie.
De bedoeling is dat de gebruiker meer controle krijgt over de huisstijl door extra kleuren toe te voegen en onderscheid te maken tussen kopteksten en body teksten (zie ook het ontwerp). Deze huisstijl elementen moeten nieuwe invoervelden krijgen in de admin panel en ze moeten zichtbaar worden in zowel de real-time editor als de edities zelf.

Model - Entity

Eerst moeten de nieuwe velden en opties in de entiteit (het model) gedefinieerd worden.

Alle mogelijke font weights worden als nieuwe optie toegevoegd. Omdat dit statische waarden zijn, die niet kunnen veranderen, wordt er ook een nieuwe array (lijst) als constante variabele aangemaakt. Hierin worden de verschillende opties opgesomd als combinatie van een waarde met een bijbehorend label.

```
1  public const DEFAULT_FONT_WEIGHT = 400;
2
3  public const FONT_WEIGHTS = [
4      self::DEFAULT_FONT_WEIGHT => 'Regular - 400',
5      100 => 'Thin - 100',
6      200 => 'Extra-light - 200',
7      300 => 'Light - 300',
8      500 => 'Medium - 500',
9      600 => 'Semi-bold - 600',
10     700 => 'Bold - 700',
11     800 => 'Extra-bold - 800',
12     900 => 'Black - 900'
13     ];
```

Vervolgens wordt er voor elk veld wordt een variabele aangemaakt, waarin de geselecteerde waarden worden opgeslagen. Boven deze variabelen wordt een ORM verwijzing geplaatst, zodat deze wijzigingen door Doctrine ook in de database worden weerspiegeld.Daarnaast worden bijbehorende getter- en setter functies aangemaakt, zodat deze waardes uitgelezen en overschreven kan worden.

Onderstaand voorbeeld geeft enkel de wijzigingen weer die zijn gemaakt voor de font-weight, maar voor elke eigenschap, zoals kleur en lettertype, zijn er op deze manier velden aangemaakt.

```php
/**
 * @ORM\Column(type="string", nullable=true)
 */
private $headingFontWeight = self::DEFAULT_FONT_WEIGHT;

/**
 * @ORM\Column(type="string", nullable=true)
 */
private $bodyFontWeight = self::DEFAULT_FONT_WEIGHT;

    public function getHeadingFontWeight(): ?string
{
    return $this->headingFontWeight;
}

public function setHeadingFontWeight(?string $fontWeight): self
{
    $this->headingFontWeight = $fontWeight;

    return $this;
}

public function getBodyFontWeight(): ?string
{
    return $this->bodyFontWeight;
}

public function setBodyFontWeight(?string $fontWeight): self
{
    $this->bodyFontWeight = $fontWeight;

    return $this;
}
```

Controller - Form

Vervolgens worden de velden aan een formulier toegevoegd. Dit soort logica bevind zich gebruikelijk in de controller, maar Symfony werkt met aparte Form types, waarin deze velden kunnen worden ondergebracht, omdat controllers idealiter zo weinig mogelijk logica bevatten. Symfony bevat een form builder waarin de velden van een formulier gekoppeld en geconfigureerd kunnen worden. Symfony zorgt vervolgens zelf achter de schermen voor de data binding; Dit houdt in dat de invoer van de gebruiker wordt verwerkt en doorgespeeld van de front end naar de back end van de applicatie.
Er wordt een variabele uit de entiteit meegegeven en het type input.
In het onderstaande voorbeeld worden er ChoiceType types gekoppeld; Dit zijn dropdown selectors.
Onder 'choices' wordt de array met font weights ingeladen, waaruit de gebruiker zou kunnen kiezen.

```
1  $builder
2  ->add('headingFontWeight', ChoiceType::class, [
3          'required' => false,
4          'label' => 'publication.form.font_weight',
5          'choices' => array_combine($options['fontWeights'],
   array_keys($options['fontWeights'])),
6          'attr' => [
7              'placeholder' => 'none',
8          ],
9      ])
10     ->add('bodyFontWeight', ChoiceType::class, [
11         'required' => false,
12         'label' => 'publication.form.font_weight',
13         'choices' => array_combine($options['fontWeights'],
   array_keys($options['fontWeights'])),
14         'attr' => [
15             'placeholder' => 'none',
16         ],
17     ])
```

View - Template

Als laatste moeten de wijzigingen ook zichtbaar zijn voor de gebruiker.

De formulier velden worden toegevoegd in de template van de admin panel om uiteindelijk door de controller gerendered te worden. Het gehele formulier zou in een keer ingeladen kunnen worden met de `{{ form(form) }}` functie van Symfony, maar daarmee gaat de mogelijkheid om het formulier te stylen verloren.
In plaats daarvan wordt elk veld apart benoemd en omgeven door custom structuur en css classes.

```
<div class="col-md-8">
    {{ form_row(form.name) }}
    <div class="d-flex flex-row color-settings mt-5">
        <div class="mr-4"> {{ form_row(form.primaryColor) }} </div>
        <div class="mr-4"> {{ form_row(form.secondaryColor) }} </div>
        <div class="mr-4"> {{ form_row(form.accentColor) }} </div>
        <div class="mr-4"> {{ form_row(form.errorColor) }} </div>
    </div>
</div>

    ...

<div class="row">
    <div class="col-md-6">
        <div class="text-settings">
            <h5 class="mb-3">{{ 'publication.form.heading' | trans }}</h5>
            <div class="row">
                <div class="col-md-12">{{ form_row(form.headingFontFamily) }}</div>
            </div>
            <div class="row">
                <div class="col-md-6">{{ form_row(form.headingTextColor) }} </div>
                <div class="col-md-6">{{ form_row(form.headingFontWeight) }} </div>
            </div>
        </div>
    </div>
    <div class="col-md-6">
        <div class="text-settings">
            <h5 class="mb-3">{{ 'publication.form.body_text' | trans }}</h5>
            <div class="row">
                <div class="col-md-7">{{ form_row(form.bodyFontFamily) }} </div>
                <div class="col-md-5">{{ form_row(form.fontSize) }}</div>
                <div class="col-md-6">{{ form_row(form.bodyTextColor) }} </div>
                <div class="col-md-6">{{ form_row(form.bodyFontWeight) }} </div>
            </div>
        </div>
    </div>
</div>
```

Omdat de nieuwe huisstijl elementen ook zichtbaar moeten worden aan de voorkant, worden deze toegepast op de templates van de editor en artikelen binnen de style tags in de head.
Dit is niet een gebruikelijke manier van het configureren van styling, omdat hiervoor bijna uitsluitend aparte css stylesheets worden gebruikt. Maar omdat er geen verbinding mogelijk is tussen deze stylesheets en de dynamische input van gebruikers, is dit een valide oplossing.

```
1  {% set headingFontFamily = edition.publication.headingFontFamily ??
   constant('DEFAULT_FONT_FAMILY', edition.publication) %}
2  {% set bodyFontFamily = edition.publication.bodyFontFamily ??
   constant('DEFAULT_FONT_FAMILY', edition.publication) %}
3  {% set headingFontWeight = edition.publication.headingFontWeight ??
   constant('DEFAULT_FONT_WEIGHT', edition.publication) %}
4  {% set bodyFontWeight = edition.publication.bodyFontWeight ??
   constant('DEFAULT_FONT_WEIGHT', edition.publication) %}
5  {% set headingTextColor = edition.publication.headingTextColor ??
   constant('DEFAULT_TEXT_COLOR', edition.publication) %}
6  {% set bodyTextColor = edition.publication.bodyTextColor ??
   constant('DEFAULT_TEXT_COLOR', edition.publication) %}
7  {% set fontSize = edition.publication.fontSize ??
   constant('DEFAULT_FONT_SIZE', edition.publication) %}
8
9  <style>
10     @import url('https://fonts.googleapis.com/css2?family={{
   headingFontFamily
   }}:wght@100;200;300;400;500;600;700;800;900&display=swap');
11     @import url('https://fonts.googleapis.com/css2?family={{ bodyFontFamily
   }}:wght@100;200;300;400;500;600;700;800;900&display=swap');
12
13     .texteditor h1, .texteditor h2, .texteditor h3, .texteditor h4,
   .texteditor h5, .texteditor h6  { font-family: '{{ headingFontFamily }}',
   sans-serif !important; font-weight: {{ headingFontWeight }} !important;
   color: {{ headingTextColor }} !important;}
14     .texteditor p { font-family: '{{ bodyFontFamily }}', sans-serif
   !important; font-weight: {{ bodyFontWeight }} !important; color: {{
   bodyTextColor }} !important;}
15
16 </style>
```

> De opmaak van de templates is erg specifiek en bevat weinig bijzonderheden en zal dus niet worden behandeld in dit document. Zie het document in de bijlage van alle wijzigingen om meer te weten te komen.

Aanmaken Call-to-action

De tweede wijziging die is doorgevoerd, is het toevoegen van ondersteuning voor call-to-actions. Deze nieuwe artikelen wijken af van de bestaande artikelen, omdat ze een knop bevatten met een achterliggende link. De gebruiker hoeft enkel de tekst en link van deze knop kunnen aanpassen, de styling is op basis van de ingestelde huisstijl.

Model - Entity

De nieuwe velden voor de knoptekst en de link van de knop worden in de algemene 'artikel' entiteit gedefinieerd.
Alle verschillende artikelen maken gebruik van deze entiteit. Er worden wederom nieuwe variabelen inclusief ORM-verwijzingen en getter- en setters functies toegevoegd.

```php
/**
 * @ORM\Column(type="text", nullable=true)
 */
private $buttonText;

/**
 * @ORM\Column(type="text", nullable=true)
 */
private $buttonLink;

public function getButtonText(): string
{
    return $this->buttonText;
}

public function setButtonText($buttonText): self
{
    $this->buttonText = $buttonText;

    return $this;
}

public function getButtonLink(): string
{
    return $this->buttonLink;
}

public function setButtonLink($buttonLink): self
{
    $this->buttonLink = $buttonLink;

    return $this;
}
```

Controller - Form

De velden aan een formulier toegevoegd. Voor de button link wordt er UrlType toegevoegd; Dit is een aangepast tekstveld die controleert of de ingevoerde tekst een geldige url is.
Voor de button text wordt er geen type toegevoegd, omdat de invoer via een custom editor gebeurt en er dus geen invoerveld hoeft weergegeven te worden op de pagina.

```php
foreach ($options['fields'] as $field => $label) {
    $fieldOptions = [
        'required' => false,
        'label' => $label,
    ];

    switch ($field) {
        ...
        case 'buttonText':
            $fieldType = null;
            break;
        case 'buttonLink':
            $fieldType = UrlType::class;
            $fieldOptions['attr'] = [
                'placeholder' => 'https://your_url.com',
                'label' => false,
                'required' => true,
            ];
            break;
       ...
    }
    $builder->add($field, $fieldType, $fieldOptions);
}
```

View

Als laatste wordt er een nieuwe configuratie aangemaakt voor de bovengenoemde editor, Quill editor. Dit is een inline javascript editor voor het wijzigen van content.
Deze editor heeft een toolbar met veel verschillende opties, maar omdat alleen de tekst van een knop gewijzigd hoeft te worden, hoeft deze toolbar niet zichtbaar te zijn.
Er is daarom een nieuwe configuratie aangemaakt, die deze balk niet weergeeft wanneer de editor een `--no-toolar` class bevat.

```
<script>
    var texteditors = document.querySelectorAll('.texteditor');

    if(texteditors !== null){
        texteditors.forEach(function (editor){
            if(!(editor.classList.contains('--skipper'))){
                if(editor.classList.contains('--no-toolbar')){
                    initQuillEditor(editor, ['link'], 'Pas knoptekst aan');
                } else {
                    initQuillEditor(editor);
                }
            }
        });
    }
</script>
```

> De opmaak van de templates is erg specifiek en bevat weinig bijzonderheden en zal dus niet worden behandeld in dit document. Zie het document in de bijlage van alle wijzigingen om meer te weten te komen.

Toevoegen templates

De derde en laatste wijziging, is het aanmaken van call-to-action artikelen zelf.
Behalve het wijzigen van de code moet er ook een preview afbeelding toegevoegd worden voor het selectiescherm van het artikel type.

Model - Entity

Er worden dit keer geen nieuwe variabelen aangemaakt, omdat alle artikelen dezelfde entiteit gebruiken.

In deze entiteit moet het nieuwe artikel type wel geregistreerd worden, zodat deze in de applicatie gebruikt kan worden.
Het artikel moet worden toegevoegd aan een array (lijst) van artikelen, er moet aangegeven worden of het een inline of standalone editor betreft en alle benodigde velden moeten worden geregistreerd.
Deze data wordt allemaal automatisch gekoppeld via naamgeving.

```
public const TEMPLATES = [
    ...
    'cta_right_background_image_left' => 'cta_right_background_image_left',
];

public const CONTENTUPLOADERTYPES = [
    ...
    'cta_right_background_image_left' => 'inline',
];
```

```
public const CTA_LEFT_BACKGROUND_IMAGE_RIGHT_FIELDS = [
    'content' => 'page.form.content',
    'imageFile' => 'page.form.image',
    'buttonLink' => 'page.form.buttonLink',
    'buttonText' => 'page.form.buttonText',
];

public const CTA_RIGHT_BACKGROUND_IMAGE_LEFT_FIELDS =
self::CTA_LEFT_BACKGROUND_IMAGE_RIGHT_FIELDS;
```

Controller - Form

> Aan de controller zijn geen noemenswaardige wijzigingen doorgevoerd, omdat deze gestandaardiseerde logica bevat voor alle artikelen. Wanneer er nieuwe artikelen worden toegevoegd, zouden deze zich moeten houden aan bestaande conventies.

View

Er worden twee views aangemaakt:

Een voor de editor van het artikel...

```
<div class="contentbox flex-column justify-content-center align-items-start">
    ...
    <div class="button-holder">
        {{ form_row(form.buttonLink) }}
        <button class="cta-button pd-background--primary --bordered" disabled>
            {{ form_widget(form.buttonText, {'attr': {'class': '--hidden'}}) }}
            <div class="texteditor --no-toolbar p-0" data-content="{{ form.buttonText.vars.value }}"></div>
        </button>
    </div>
</div>
```

...en een voor de online weergave van het artikel

```
<div class="pd-background--primary--static py-1 mb-4">
      <div class="content-box --fullcenter">
            <div class="content-wrapper">
                <div data-aos="fade-up">
                    {{ page.content | replace(replaceParams) | raw }}
                    <button class="mt-5 btn cta-button pd-background--primary --bordered"> {{ page.buttonText | replace(replaceParams) | raw }} </button>
                </div>
            </div>
      </div>
</div>
```

Beide templates dienen apart opgemaakt te worden met scss styling.

> De opmaak van de templates is erg specifiek en bevat weinig bijzonderheden en zal dus niet worden behandeld in dit document. Zie het document in de bijlage van alle wijzigingen om meer te weten te komen.