Translation Management Diagram: 8/19/2021

Current Implementation: Static Data Via React Widget

Editor Publishes Translated Pages

- page must have url with language code (-esp) suffix for language
- If the the main page url or any translated page urls change, then widget will break until it's code has been updated accordingly.

↓

Ticket is created for FE Widget Update with new language page slugs

- Need urls of newest translated content for each language
- **All** languages need to be published for widget to be 'turned on' . If all language slugs aren't available/published then language selector will malfunction.

↓

Developer Submits PR For New Pages

- Unit tests and JSON/constants file is updated

↓

After PR approval and content build, then page goes live.

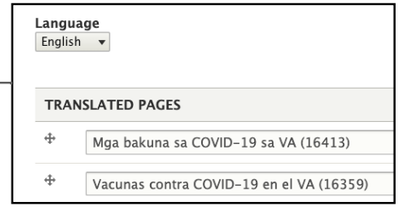
- Unit tests and JSON/constants file is updated

Proposed Implementation 1: Language Field and Entity Reference Field

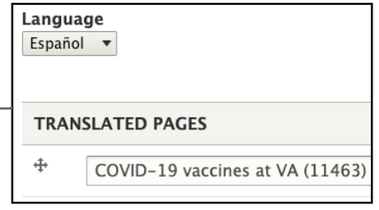
Editor Publishes Translated Pages

- page URL does not matter
- the language field must be set
- on the translated page, the englisg (parent) page must be listed in the translated pages field list.
- on the english page all other translated pages should be listed in the translated pages field list

English page



Translated Page



↓

After pages are published with relationships, the content-build proces injects relevant page data

- React widget parses a data-translated-pages attribute and determines whether to show I18Select component or not on the page.
- Only *published* translated pages that have been referenced will show up, so a translation in draft will not have a link displayed for it.
- Not all lanuages have to be show and the react widget will display if there are a minumum of 2 languages available for a page (the english and at least one non-english translation)
- No PR or code change to the react widget would be needed aside from the addition of a whole new language set being added to the platform translations.

Proposed Implementation 2: Drupal Multilingual Module Usage

Editor Publishes Page and Translated Content That is Attached to That Node

- each language is part of the single node data struture
- there are not seperate pages for each language
- no reference field is needed
- there is a different language field that is exposed with the multilingual modules, so that field would need to be used instead of the dropdown field example from implementation 2

↓

After a page and it's corresponding language data is published, then a process is used to translate content on the frontend.

- Our current content-build process would have to be reworked to query languages for nodes and generate these pages when present.
- If we move to a public drupal content api, then a react component may call an endpoint for translated content and handle that directly from the frontend.
- No POC or method forward has been created for this implementation yet, and most likely would be best suited for once a public content api is available.