Translator API - An API with several endpoints to manage glossaries and translations.

Use this test to demonstrate your understanding of OO and TDD.

Task:
- Implement an API-only Ruby on Rails application.
- Use JSON as the response format.
- Provide tests for your code.
- You can use any sql database of your choice.
- You can use any gems you like.
- Send us your zipped solution.

Required endpoints:

`POST /glossaries`
Description: Creates a glossary.
Params:
`source_language_code` - an ISO 639-1 source language code
`target_language_code` - an ISO 639-1 target language code
Requirements:
- a glossary must be unique within the system (scoped to language codes)
- language code fields must contain a valid ISO 639-1 code
- the list of possible ISO 639-1 codes is included in the file

`GET /glossaries/<id>`
Description: Returns the glossary with all terms.

`GET /glossaries`
Description: Returns all glossaries with terms.

`POST /glossaries/<id>/terms`
Description: Creates a term in the given glossary.
Params:
`source_term` - a source term
`target_term` - a target term
Requirements:
- source and target terms cannot be blank

`POST /translations`
Description: Creates a translation. Note: the translation process will not take place within this application.
Params:
`source_language_code` - an ISO 639-1 source language code
`target_language_code` - an ISO 639-1 target language code

`source_text` - a source text to be translated
`glossary_id` - an existing glossary id

Requirements:

- `source_language_code, target_language_code, source_text` fields must be present
- the `glossary_id` is optional
- if the `glossary_id` is provided `source_language_code` and `target_language_code` for translation and glossary must match
- the maximum length of the source text is 5000 characters

`GET /translations/<id>`

Description: Returns the translation.

Requirements:

- if the glossary is present the `source_text` should contain highlighted fragments representing the glossary terms (if found)
- use `<HIGHLIGHT></HIGHLIGHT>` tags to represent highlights in the `source_text`
- include a list of matching terms in the response

Example:

source_text: `"This is a recruitment task."`
glossary term: `"recruitment"`
highlighted source_text: `"This is a <HIGHLIGHT>recruitment</HIGHLIGHT> task."`

Bonus:

- Provide a Dockerfile which allows to run your app.