Middleware development

# Purpose

This was actually, by far, the hardest part. I’d already written a very simple middleware for getting the default language of the user’s browser from the accept-language header. However it felt too simplistic, even for an assignment we only had a week to complete. But on my way home from class on Wednesday, it hit me: The perfect middleware for this project would be a translation one. From that realization, the design and role was clear; I needed middleware that could batch translate.

# Design

The design was easy: I just needed to go from one to many, I.E batch translate. But obviously that wasn’t enough, so the middleware can handle a number of sentences in an array and will translate all of them into multiple languages. So the relationship is actually many to many. I also realized that it would need to send the translation results back to the client for human verification and edits.

# Planning process

The reason I chose to implement this specific middleware was because it will allow me (and other superuser(s)) to quickly add new words and/or grammar to Dialogo. Obviously, Google Translate can’t be trusted 100%, so the translations will still have to be checked manually. See above how the middleware handles this.

# Integration details

I added a “/translations” route for the middleware to the server. That’s literally it. I will also eventually have to make a proper client interface to enable batch translations for superusers, but for now it will be very basic.

The server will eventually have to support arguments from the client specifying what kind(s) of translations these are, but these arguments will most likely be parsed by the route handler for propagation into the DB and not by the middleware itself.

# Usage examples

import express from "express";

import HttpCodes from "../modules/httpErrorCodes.mjs";

import Translate from "../middleware/Translate.js";

const TRANSLATION\_API = express.Router();

TRANSLATION\_API.post("/", Translate.translateFrom, (req, res) => {

    //TODO: Propagate translations into DB...

});

/\*\*A very simple middleware for handling errors

 \* that sends a generic error message to the client.

 \*/

TRANSLATION\_API.use((err, req, res, next) => {

    console.error(err);

    res.status(HttpCodes.ClientSideErrorResponse).send('An error occurred.');

    next();

})

export default TRANSLATION\_API;