# Blogitekstin kääntäminen englanniksi

## Projektin kuvaus

Projektin tarkoituksena on kääntää blogitekstin englanniksi käyttäen tekoälyä. Luodaan nappi jota painamalla tekoäly käy läpi blogin jokaisen artikkelin ja kääntää sen. Käännös tallennetaan tietokantaan ja tulostetaan näytölle. Tarkoituksena on ensin käydä tarkastamassa tietokannasta että onko teksti jo käännetty ennen kuin annetaan tekoälylle käännös-käsky. Näin säästetään aikaa ja rahaa. Englanti-napin viereen tulee myös Suomi-nappi jota painamalla alkuperäinen teksti tulee näkyviin.

Tekoälynä käytetään ChatGPT 4o-mini.

## Tietokanta-taulukon luominen

Luodaan tietokantaan taulu ARTICLE_TRANSLATIONS jonne tallennetaan käännettävät tekstit:

```
CREATE TABLE `ARTICLE_TRANSLATIONS` (
  `translation_id` int NOT NULL,
  `article_id` int NOT NULL,
  `source_hash` char(64) DEFAULT NULL,
  `lang_code` varchar(10) NOT NULL,
  `translated_title` text NOT NULL,
  `translated_content` text NOT NULL,
  `created_at` timestamp NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

- **translation_id:** Pääavain jokaiselle käännösriville. Yksilöllinen tunniste. Yleensä asetetaan automaattisesti kasvavaksi (AUTO_INCREMENT).
- **article_id:** Viittaa alkuperäisen artikkelin `ARTICLE.article_id`-sarakkeeseen (vierasavain). Yhdistää käännöksen oikeaan artikkeliin.
- **source_hash:** SHA-256-tarkiste (hash), joka lasketaan artikkelin otsikosta ja sisällöstä. Käytetään tarkistamaan, onko sisältö muuttunut käännöksen jälkeen.
- **lang_code:** Kielen tunniste (esim. `en`, `fi`, `sv`). Mahdollistaa useiden kieliversioiden tallentamisen samaan tauluun.
- **translated_title:** Käännetty artikkelin otsikko.
- **translate_content:** Käännetty artikkelin sisältö.
- **created_at:** Automaattisesti luotu aikaleima, joka kertoo milloin käännös tallennettiin.
- **ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;**
    - **ENGINE = InnoDB:** Mahdollistaa vierasavaimet (`FOREIGN KEY`) ja transaktiot.
    - **utf8mb4:** Tukee myös Unicode-emojit ja erikoismerkit.

- **COLLATE = utf8mb4_0900_ai_ci:** Määrittää aakkosjärjestyksen ja vertailutavan (case-insensitive).

# Käännös-painikkeen luominen

Luodaan nappi, jota painamalla suomeksi kirjoitettu teksti voidaan kääntää englanniksi. Nappi luodaan tiedostoon joka on tallennettu tietokantaan.

```
<!--30.9.25 Luodaan painike jonka avulla teksti käännetään englanniksi -->
<div class="translate-container">
  <form
    hx-post="../verifications/translate.php"
    hx-target=".article-container"
    hx-swap="innerHTML"
    hx-indicator="#loading-indicator"
   >
     <input type="hidden" name="slug" value="{blog_slug}">
     <button type="submit">EN</button>
   </form>
   <button type="button" onclick="location.reload()">FI</button>
   <div id="loading-indicator" class="htmx-indicator">
      Translating...
   </div>
</div>
```

- Luodaan translate-container jossa on lomake ja painikkeet.
- Luodaan lomake jossa on HTMX -toiminnot:
    - **hx-post:** lähetetään lomakkeen tiedot translate.php tiedoston käsiteltäväksi.
    - **hx-target:** näytetään translate.php käsiteltävät tiedot article-containerissa (jonne tulostetaan generoitavat blogipostaukset).
    - **hx-swap:** innerHTML -vaihdetaan vain article-containerin sisältö, muu HTML jää entiselleen.
    - **hx-indicator:** näytetään ilmoitus että käännöstä suoritetaan. Ilmoitus näkyy sen ajan mikä translate.php tiedosto käyttää.
- `<input type="hidden" name="slug" value="{blog_slug}">`
    - Annetaan tieto blog_slug mukaan translate.php:lle.
- Painike “EN” käynnistää lomakkeen siirron.
- Painike “FI”: kun painiketta painaa, refressataan sivusto jolloin alkuperäinen, suomenkielinen teksti palautuu.

# Käännöstiedosto translate.php

## Tiedoston toiminnot

### Tiukka tyyppimääritys - declare(strict_types)

Varmistetaan että PHP käsittelee tiukasti tyypitettyjä arvoja. PHP heittää virheen jos annetaan väärän tyyppisiä arvoja.

```
declare(strict_types=1);
```

### Turva-otsakkeiden lataus

```
require_once '../private/security_headers.php';
```

Turva-otsake-tiedosto:

```
<?php
// Security headers
// Estetään Clickjacking-hyökkäykset.
header("X-Frame-Options: DENY");
// Estetään selainta arvaamasta tiedostotyyppiä.
header("X-Content-Type-Options: nosniff");
// Viittaajatieto lähetetään vain, jos linkin kohde on samalla domainilla.
header("Referrer-Policy: same-origin");
// Yrittää estää perinteisiä XSS-hyökkäyksiä selaimen omalla suodatuksella.
header("X-XSS-Protection: 1; mode=block");
// Määritellään tarkasti, mistä lähteistä selain saa ladata resursseja.
header("Content-Security-Policy: default-src 'self'; img-src 'self' data:;
connect-src 'self' https://api.openai.com;");
```

### Kapseli-tiedoston haku

Haetaan istunnonhallinta, tietokantakyselyt ja funktiot

```
require_once '../private/bootstrap.php';
```

Kapseli-tiedosto:

```
<?php
// Kapselointia varten tehty tiedosto
require_once 'config.php'; // Virheen käsittely

// Aloitetaan sessio jos se ei vielä ole aloitettu
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}
```

```php

// Tietokantakyselyt
require_once '../database/db_connect.php';
require_once '../database/db_enquiry.php';
require_once '../database/db_add_data.php';

// Funktioiden käsittely
require_once '../funcs.php';
```

## Composer autoloader lataus

```php
require __DIR__ . "./../vendor/autoload.php";
```

## Dotenv ja OpenAI -kirjastot

```php
use Dotenv\Dotenv;
use OpenAI\Client;
```

## Ladataan .env-tiedosto vain lokaalisti (jos se on olemassa)

```php
if (file_exists(dirname(__DIR__, 1) . '/.env')) {
    $dotenv = Dotenv::createImmutable(dirname(__DIR__, 1));
    $dotenv->load();
}
```

## API-avaimen haku turvallisesti ympäristömuuttujista

```php
$apiKey = getenv('OPENAI_API_KEY') ?: ($_ENV['OPENAI_API_KEY'] ?? null);
if (!$apiKey) {
    http_response_code(500);
    error_log("OpenAI API key missing");
    exit("Järjestelmävirhe, yritä myöhemmin uudelleen");
}
```

## Luodaan OpenAI-client

```php
$client = OpenAI::client($apiKey);
```

## Rate-limit asetus ja tarkistus

```php
$minDelayBetweenRequests = 3; // sekuntia
$lastRequestTime = (int)($_SESSION['last_translate_time'] ?? 0);
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    header("Location: ../index.php");
    exit();
}
```

```php
// Rate-limit tarkistus
if(time() - $lastRequestTime < $minDelayBetweenRequests) {
    http_response_code(429);
    exit("Liian monta pyyntöä, odota hetki ja yritä uudelleen.");
}
$_SESSION['last_translate_time'] = time();
```

- Määritellään että kuinka monta sekuntia täytyy kulua ennen kuin käyttäjä voi tehdä seuraavan käännöspyynnön.
- Haetaan istunnosta aika, jolloin käyttäjä viimeksi teki käännöspyynnön. Oletus 0.
- Sallitaan vain POST-pyynnöt.
- Lasketaan aika edellisestä pyynnöstä, jos se on pienempi kuin sallittu väli, pyyntö hylätään.
- Tallennetaan nykyinen pyynnön aikaleima, jotta seuraava pyyntö voi tarkistaa sen.

## Slug validointi

```php
$rawSlug = $_POST['slug'] ?? '';
if (!is_string($rawSlug) || $rawSlug === '') {
    http_response_code(400);
    exit("Virheellinen pyyntö: slug puuttuu.");
}
// Salli vain a-ö, 0-9 ja viiva/underscore
if(!preg_match('/^[a-ö0-9\-_.]+$/i', $rawSlug)) {
    http_response_code(400);
    exit("Virheellinen slug.");
}
$slug = $rawSlug; // Turvallinen käyttää prepared statementissa
```

## Haetaan artikkelit tietokannasta

```php
try {
    $stmt = $conn->prepare("SELECT article_id, title, content, image_path
                            FROM ARTICLE
                            WHERE blog_id = (SELECT blog_id FROM BLOG WHERE
slug = ? LIMIT 1)
                            AND status='PUBLISHED'
                            AND deleted_at IS NULL
                            ORDER BY published_at DESC");
    $stmt->bind_param("s", $slug);
    if ($stmt === false) {
        throw new RuntimeException("Prepare failed: " . $conn->error);
    }
    $stmt->execute();
```

```
    $result = $stmt->get_result();
    $articles = $result ? $result->fetch_all(MYSQLI_ASSOC) : [];
    $stmt->close();
} catch (Throwable $e) {
    error_log("[DB] Failed fetching articles for slug={$slug}: " .
$e->getMessage());
    http_response_code(500);
    exit("Palvelinvirhe – yritä myöhemmin uudelleen.");
}
```

## Määritellään kieli

```
$lang = "en";
```

## Artikkelien iterointi

```
foreach ($articles as $article) {
```

### Tyyppitarkistukset

```
$article_id = isset($article['article_id']) ? (int)$article['article_id'] : 0;
$title_raw = (string)($article['title'] ?? '');
$content_raw = (string)($article['content'] ?? '');
$image_path_raw = isset($article['image_path']) ?
(string)$article['image_path'] : '';
```

### Lasketaan hash (otsikko + sisältö)

```
$source_hash = hash('sha256', $title_raw . "\n" . $content_raw);
```

### Tarkistetaan onko käännös jo tallennettu ja vastaako hashia

```
try {
        $checkStmt = $conn->prepare("SELECT translated_title,
translated_content, source_hash
                                     FROM ARTICLE_TRANSLATIONS
                                     WHERE article_id = ? AND lang_code = ?");
        if ($checkStmt === false) {
            throw new RuntimeException("Prepare failed: " . $conn->error);
        }
        $checkStmt->bind_param("is", $article_id, $lang);
        $checkStmt->execute();
        $checkResult = $checkStmt->get_result();
    } catch (Throwable $e) {
        error_log("[DB] Failed checking translations: " . $e->getMessage());
        // Jos tietokantavirhe, hyppää seuraavaan artikkeliin
```

```
        continue;
    }
```

Alustetaan käännösmuuttujat

```
$translatedTitle = '';
$translatedContent = '';
```

Artikkeleiden hakua ja hashin vertailu

```
    if ($checkResult && $checkResult->num_rows > 0) {
        $translationRow = $checkResult->fetch_assoc();
        // Verrataan tietokannassa olevaan hashiin käännettyä hashia
        if (isset($translationRow['source_hash']) &&
$translationRow['source_hash'] === $source_hash) {
            // Sama hash -> käytetään olemassa olevaa käännöstä
            $translatedTitle = $translationRow['translated_title'];
            $translatedContent = $translationRow['translated_content'];
} else {
// Hash muuttui -> käännetään uudelleen
$translatedTitle = translateWithOpenAI($client, $title_raw);
$translatedContent = translateWithOpenAI($client, $content_raw);

// Päivitetään käännös ja hash
try {
    $updateStmt = $conn->prepare("UPDATE ARTICLE_TRANSLATIONS
    SET translated_title=?, translated_content=?, source_hash=?
    WHERE article_id=? AND lang_code=?");
    if ($updateStmt === false) {
      throw new RuntimeException("Prepare failed: " . $conn->error);
    }
    $updateStmt->bind_param("sssis", $translatedTitle, $translatedContent,
$source_hash, $article_id, $lang);
    $updateStmt->execute();
    $updateStmt->close();
} catch (Throwable $e) {
  error_log("[DB] Failed updating translation for article_id={$article_id}: "
. $e->getMessage());
  }
}
```

Uuden käännöksen luonti

```
$translatedTitle = translateWithOpenAI($client, $title_raw);
$translatedContent = translateWithOpenAI($client, $content_raw);
```

Kutsutaan funktiota translateWithOpenAI jossa suoritetaan otsikon ja sisällön kääntäminen

```
function translateWithOpenAI($client, string $text): string {
    // Älä lähetä tyhjää
    $text = trim($text);
    if ($text === '') {
        return '';
    }

    // Luodaan pyyntö ja käsitellään virheet
    try {
        $response = $client->chat()->create([
            'model' => 'gpt-4o-mini',
            'messages' => [
                ['role' => 'system', 'content' => 'You are a translation
engine. Always translate from Finnish to English. Return only the translated
text.'],
                ['role' => 'user', 'content' => $text]
            ],
        ]);

        // Etsi turvallisesti vastaus
        $content = $response->choices[0]->message->content ?? null;
        if (!is_string($content) || $content === '') {
            error_log("[OpenAI] Empty response for translation request.");
            return $text;
        }
        // Trimmaa ja palauta
        return trim((string)$content);

    } catch (Throwable $e) {
        error_log("[OpenAI] translate error: " . $e->getMessage());
        // fallback: palauta alkuperäinen teksti, näin et paljasta
virheilmoituksia käyttäjälle
        return $text;
    }
}
```

Tallennetaan uudet käännökset tietokantaan

```
try {
            $insertStmt = $conn->prepare(
                "INSERT INTO ARTICLE_TRANSLATIONS
```

```
                (article_id, lang_code, translated_title, translated_content,
source_hash)
                VALUES (?, ?, ?, ?, ?)"
            );
            if ($insertStmt === false) {
                throw new RuntimeException("Prepare failed: " . $conn->error);
            }
            $insertStmt->bind_param("issss", $article_id, $lang,
$translatedTitle, $translatedContent, $source_hash);
            $insertStmt->execute();
            $insertStmt->close();
        } catch (Throwable $e) {
            error_log("[DB] Failed inserting translation for
article_id={$article_id}: " . $e->getMessage());
        }
    }
    // Suljetaan checkStmt
    $checkStmt->close();
```

## Renderöinti

```
$safeTitle = htmlspecialchars($translatedTitle, ENT_QUOTES | ENT_SUBSTITUTE,
'UTF-8');
    $safeContent = htmlspecialchars($translatedContent, ENT_QUOTES |
ENT_SUBSTITUTE, 'UTF-8');

    echo "<div class='article'>";
    echo "<h2>{$safeTitle}</h2>";

    if (!empty($article['image_path'])) {
        echo "<img src='" . htmlspecialchars($image_path_raw, ENT_QUOTES,
'UTF-8') . "' alt='Artikkelin kuva' class='article-image'>";
    }
    echo "<p class='article-content'>" . nl2br($safeContent) . "</p>";
    echo "</div>";
}
```