Documentación Técnica

1. Resumen

Sistema web que permite al usuario seleccionar un país y ciudad de destino, ingresar un presupuesto en pesos colombianos y visualizar:

- La conversión del presupuesto a la moneda local (con símbolo y tasa de cambio).
- Información del clima actual en la ciudad destino.

El **frontend** está desarrollado en **Angular** y consume APIs a través del **backend en Laravel**, que actúa como intermediario para obtener los datos de cambio de moneda y clima.

2. Arquitectura

```
[Usuario]
   |
[Angular (frontend)]
   | (HTTP/REST)
[Laravel API (backend)]
   |
[APIs Externas: Exchangerate.host / OpenWeatherMap]
   |
[Base de datos MySQL]
```

- **Frontend (Angular)**:
    - Componentes Step1, Step2, Step3.
    - Servicios singleton para compartir estado (TravelService).
    - Traducciones con @ngx-translate.
- **Backend (Laravel)**:

    - Controladores REST (por ejemplo HistoryController) para guardar/leer historial.
    - Conexión a MySQL usando Eloquent.
    - Llamadas a APIs externas (Exchangerate.host, OpenWeatherMap).
- **Base de datos MySQL**:
    - Tablas parametrizadas (países, ciudades, monedas).
    - Tabla historial de consultas para guardar cada búsqueda de usuario.

3. Frontend Angular

- **Step1Component**: selección de país y ciudad.
- **Step2Component**: ingreso de presupuesto y selección de moneda de conversión.
- **Step3Component**: muestra resultado de conversión y clima.
- **TravelService** (singleton): mantiene los datos seleccionados entre componentes y ejecuta llamadas HTTP al backend.
- **Internacionalización**: @ngx-translate carga archivos es.json y de.json en assets/i18n.

4. Backend Laravel

- **Rutas API** en routes/api.php, ejemplo:

```
Route::post('/history', [HistoryController::class, 'store']);

Route::get('/exchange', [ExchangeController::class, 'convert']);

Route::get('/weather', [WeatherController::class, 'current']);
```

- **Controladores**:
    - ExchangeController llama a Exchangerate.host usando Http::get().
    - WeatherController llama a OpenWeatherMap usando Http::get().

    - HistoryController guarda los datos de cada operación en la base de datos.

- **Configuración. env** para base de datos y keys API:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=infodec_db
DB_USERNAME=root
DB_PASSWORD=
EXCHANGE_API_KEY=tu_key_exchangerate
WEATHER_API_KEY=tu_key_openweather
```

5. Base de Datos

Tablas recomendadas:

- **countries** (id, name_es, name_de, code, currency_code, currency_name, currency_symbol)
- **cities** (id, country_id, name_es, name_de, lat, lon)
- **history** (id, user_id opcional, country_id, city_id, amount_cop, amount_converted, exchange_rate, created_at)

Puedes importar la estructura desde Workbench (infodec_db.sql) y correr migraciones en Laravel.

6. APIs externas

- **Exchangerate.host**
  GET
  https://api.exchangerate.host/convert?access_key=YOUR_KEY&from=COP&to=GBP&amount=100000
- **OpenWeatherMap**
  GET
  https://api.openweathermap.org/data/2.5/weather?q=Londres&units=metric&appid=YOUR_KEY

Laravel recibe estos parámetros y devuelve la respuesta al frontend Angular.

7. Seguridad y sesiones

- El frontend Angular no maneja sesiones de servidor.
- Laravel maneja las peticiones de API y puede almacenar sesiones si en un futuro se implementa autenticación de usuario.

8. Internacionalización

Se usan archivos es.json y de.json para traducciones de texto. Ejemplo:

// es.json

```
{
  "STEP1_TITLE": "Selecciona país y ciudad",
  "STEP2_TITLE": "Introduce tu presupuesto",
  "STEP3_TITLE": "Resultado de la conversión",
  "BUDGET_COP": "Presupuesto (COP)"
}
// de.json
{
  "STEP1_TITLE": "Wähle Land und Stadt",
  "STEP2_TITLE": "Gib dein Budget ein",
  "STEP3_TITLE": "Ergebnis der Umrechnung",
  "BUDGET_COP": "Budget (COP)"
}
```

9. Flujo de uso

1. El usuario selecciona país y ciudad en Step1.
2. El usuario ingresa presupuesto y selecciona moneda en Step2.
3. El sistema llama al backend:
    - Guarda historial.
    - Consulta la API de cambio de moneda y clima.
4. En Step3 se muestran:
    - País y ciudad destino.
    - Presupuesto original en COP.
    - Presupuesto convertido con símbolo.
    - Tasa de cambio.
    - Clima actual.

10. Mantenimiento

- Para agregar más países/ciudades: añadir en DB y en traducciones.
- Para cambiar API: editar métodos en Laravel.
- Para agregar autenticación: usar Laravel Sanctum o Passport.