POPULATION GROWTH RATE API

GLOSSARY

Term	Definition
CRUD	Describes the 4 operations a user should be able to do with a REST API ie, Create, Read, Update, Delete
REST	A set of guidelines describing how an API should be designed
PGR	Population Growth Rate
CORS	Cross-Origin Resource Sharing; browser security feature controlling domain-based access to APIs.
HTTP/HTTPS	HyperText Transfer Protocol (Secure); the protocol used for web requests and responses.
Actix-Web	A high-performance Rust web framework built on the Tokio async runtime.
Tokio	An asynchronous runtime for Rust, providing event-driven, non-blocking I/O.
Nivo	A rich React charting library built on top of D3.

INTRODUCTION

1.1 Purpose

This document defines the functional and technical requirements for the Population Growth Rate (PGR) Analytics API.It outlines the objectives, features, and constraints of the system in order to provide a clear roadmap for design, development, testing, and deployment.

1.2 Scope

In Scope

API Endpoints

- CRUD Operations - Create, Read, Update & Delete country population records with validations and logging.
- Return the top N countries by population growth rate.
- Delivers the average population and growth rate for each decade.
- Returns two side-by-side lists of yearly population and growth rate for each country.
- Returns the average or median population statistics based on the year or decade.

Security Considerations

- Basic API safeguards (CORS configuration, input sanitization)

Visualization

- Front-end & visualization components using React & charting

Out of Scope

Monitoring

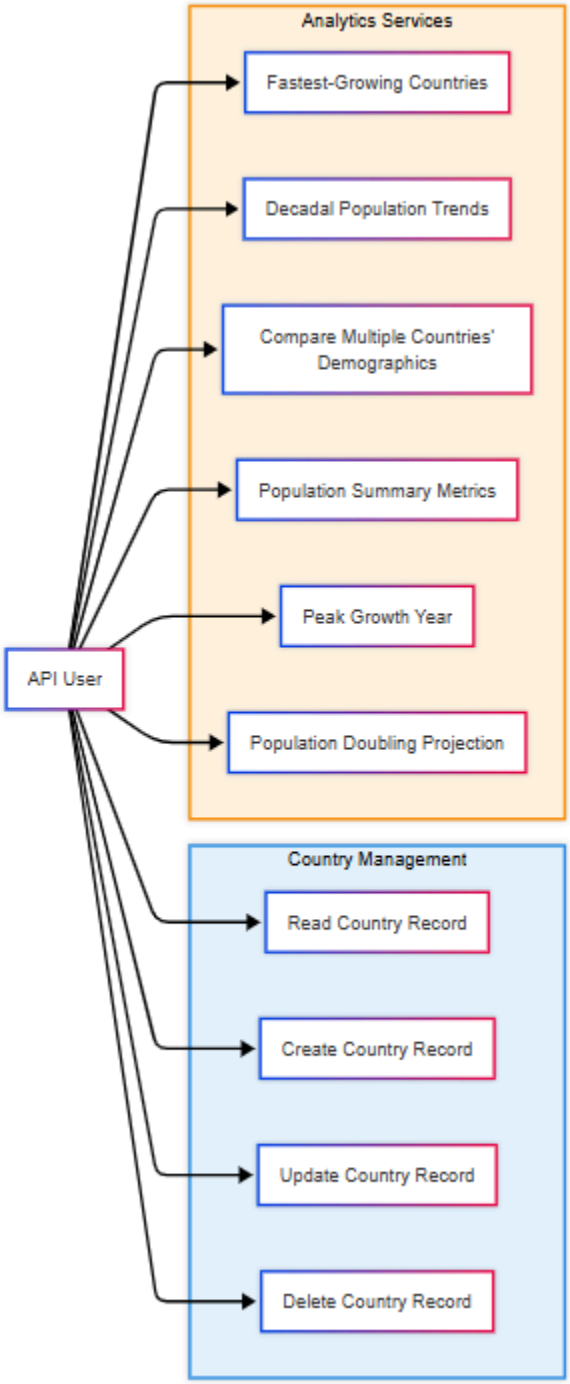
- Operational monitoring, logging infrastructure, and DevOps pipelines

Database

- Detailed database schema design beyond a high-level overview.

USE CASES

2.0 - Use Case Model



2.1 Use Case 1 - Read Country Record

Use Case	Read Country Record
Description	Retrieve a country's latest population, growth amount, growth rate, and decade by name or ISO3
User Story	As a user, I want to fetch a country's most recent demographics so I can view current data

2.2 Use Case 2 - Create Country Record

Use Case	Create Country Record
Description	Add a new country population entry
User Story	As a user, I want to add a new country's population record so the dataset stays up to date

2.3 Use Case 3 - Update Country Record

Use Case	Update Country Record
Description	Modify fields of an existing country population entry based on name or ISO3
User Story	As a user, I want to update a country's population or growth rate so corrections are applied

2.4 Use Case 4 - Delete Country Record

Use Case	Delete Country Record
Description	Remove a country's population entry from the dataset
User Story	As a user, I want to delete obsolete or incorrect records so the dataset remains accurate

2.5 Use Case 5 - Fastest Growing Countries

Use Case	Return the Fastest Growing Countries
Description	Return the top N countries by population growth rate for a specified year
User Story	As a user, I want to see which countries grew fastest in a given year so I can identify hotspots

2.6 Use Case 6 - Population Trends Over Decades

Use Case	Global Trends by Decade
Description	Show each decade's average population and growth rate, or only a specified decade's entry
User Story	As a user, I want to compare decadal population trends so I can study long-term demographic shifts

2.7 Use Case 7 - Compare Multi-Country Demographics

Use Case	Compare N number of Countries
Description	Return side-by-side yearly population and growth rate lists for multiple countries
User Story	As a user, I want to benchmark countries demographic histories to inform policy decisions

2.8 Use Case 8 - Population Summary Metrics

Use Case	Return Population Summary Metrics
Description	Compute summary metrics (mean, median, min, max) grouped by year or decade.
User Story	As a user, I want summary figures for population data

2.9 Use Case 9 - Peak Growth Years

Use Case	Calculate Peak Growth Years
Description	Identify the year and value when a country's annual growth rate was highest
User Story	As a user, I want to know when growth peaked so I can investigate its underlying causes

2.10 Use Case 10 - Population Doubling Projection

Use Case	Doubling Population Estimation
Description	Estimate how many years it would take for a country's population to double based on a given year's growth rate.
User Story	As a user, I want a doubling-time estimate so I can plan future infrastructure needs.

INTERFACES

3.1 System Interfaces

- **React Front-End** - A Single-Page Application built with React and Tailwind CSS. It communicates with the API over HTTPS to fetch data and render tables and interactive charts using Nivo.
- **Rust CLI Tool** - A command-line utility in Rust for batch queries and administrative operations against the API.

3.2 Software Interfaces

- **Actix-Web REST API** - Exposes endpoints defined in the functional requirements, implemented in Rust with the Tokio runtime.
- **Serde JSON** - Used for serialization and deserialization of request and response payloads between the client and the server.
- **Nivo Chart Components** - React chart components that consume the API's JSON responses to visualize demographic data.

3.3 Communication Interfaces

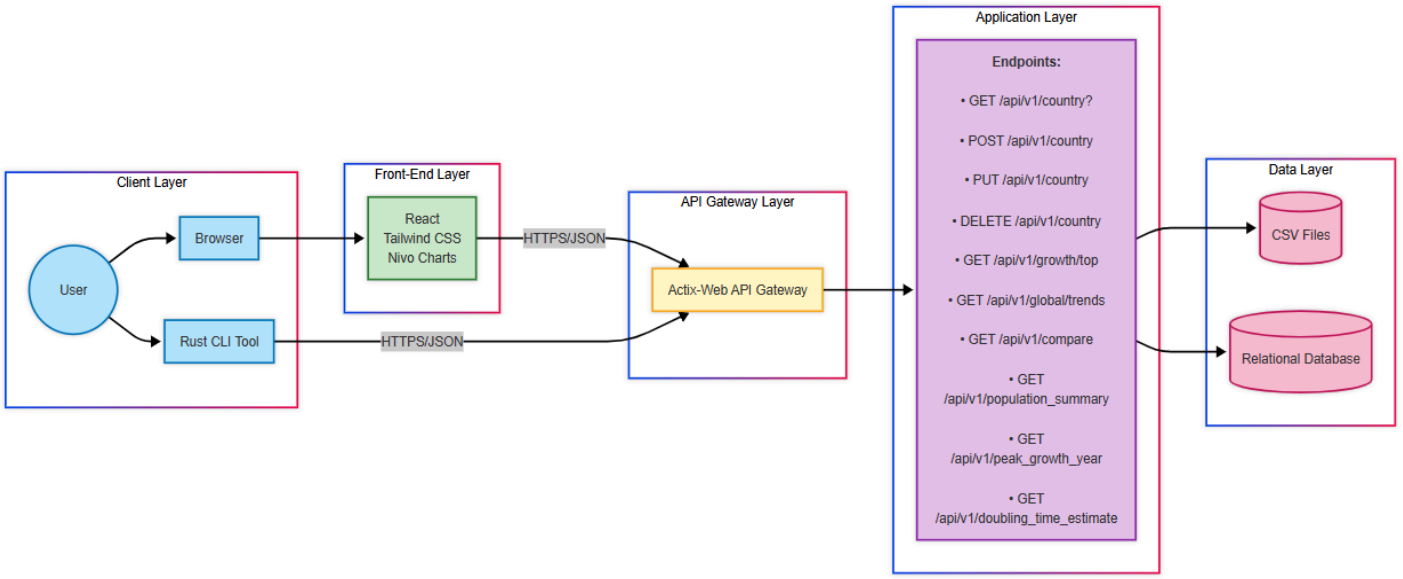
- **Data Format** - JSON for both requests and responses, following a consistent envelope pattern and appropriate HTTP status codes.
- **CORS** - Configured to allow browser-based front-ends from approved origins to call the API.

SOLUTION DESCRIPTION

4.1 Summary

The Population Growth Rate (PGR) API is a modular, API-first system designed to provide access to global demographic data. The core is a Rust-based backend built on Actix-Web and Tokio, exposing a suite of RESTful endpoints for CRUD operations, analytical queries (e.g, top-growth, decadal trends, comparisons, and summary metrics), and specialized insights.
For visualization, a React single-page application styled with Tailwind CSS will consume the API and render interactive, responsive charts using Nivo. This separation of concerns ensures clear boundaries between data services and presentation, while enabling rapid iteration and easy maintenance. Configuration, logging, and error handling follow industry best practices.

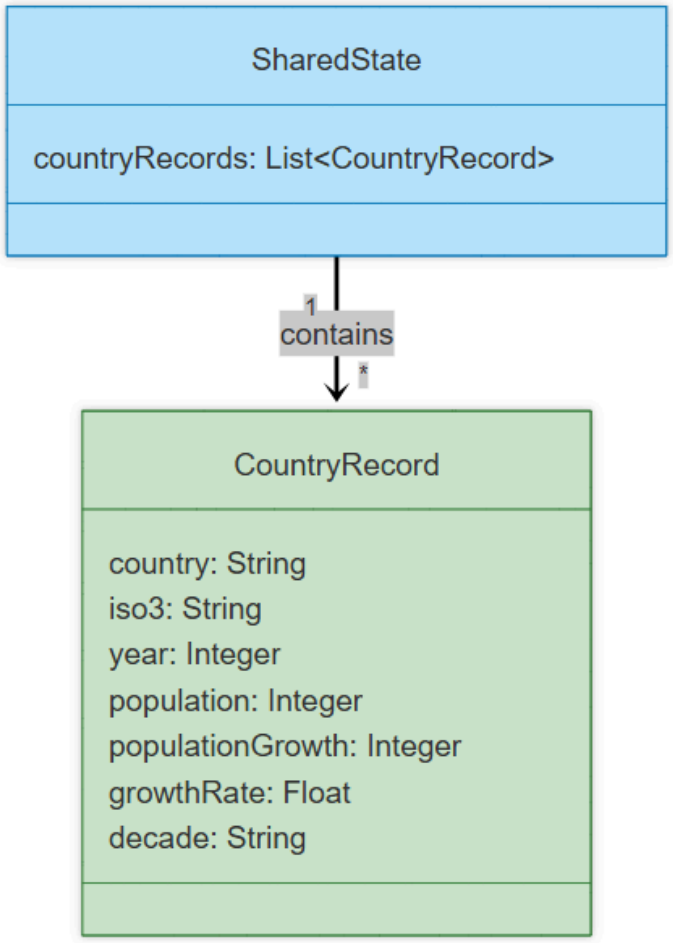
4.2 System Architecture



- Client Layer: Users interact via a web browser or a Rust CLI tool.
- Front-End Layer: A React single-page app (styled with Tailwind CSS and rendering charts with Nivo) runs in the browser and issues HTTPS/JSON calls.

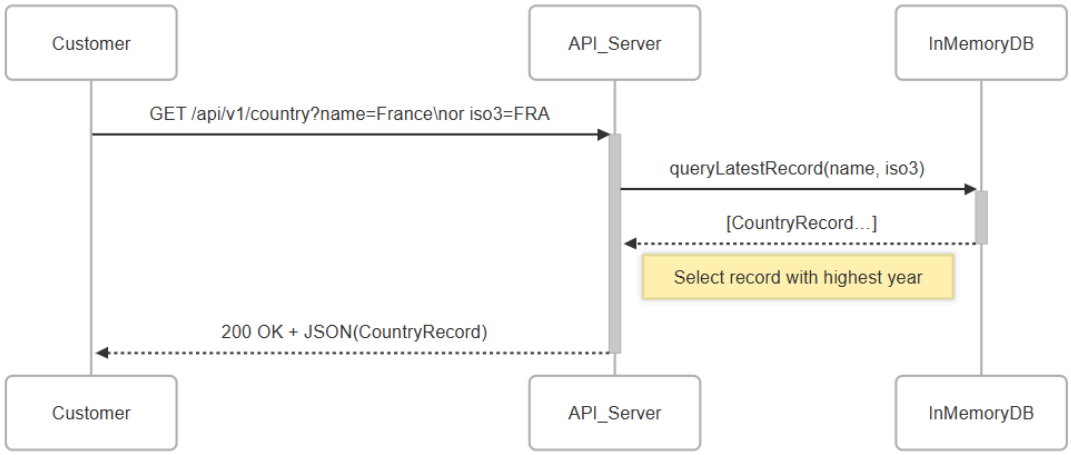
- API Gateway Layer: An Actix-Web server sits exposed at /api/v1/..., routing every request into the core application.
- Application Layer: Implements all business logic—CRUD on population records, top-growth queries, decadal trends, comparisons, summaries, peak-growth, and doubling-time endpoints.
- Data Layer: The application reads and writes demographic data from CSV files or a relational database, depending on configuration.

4.3 Database Design



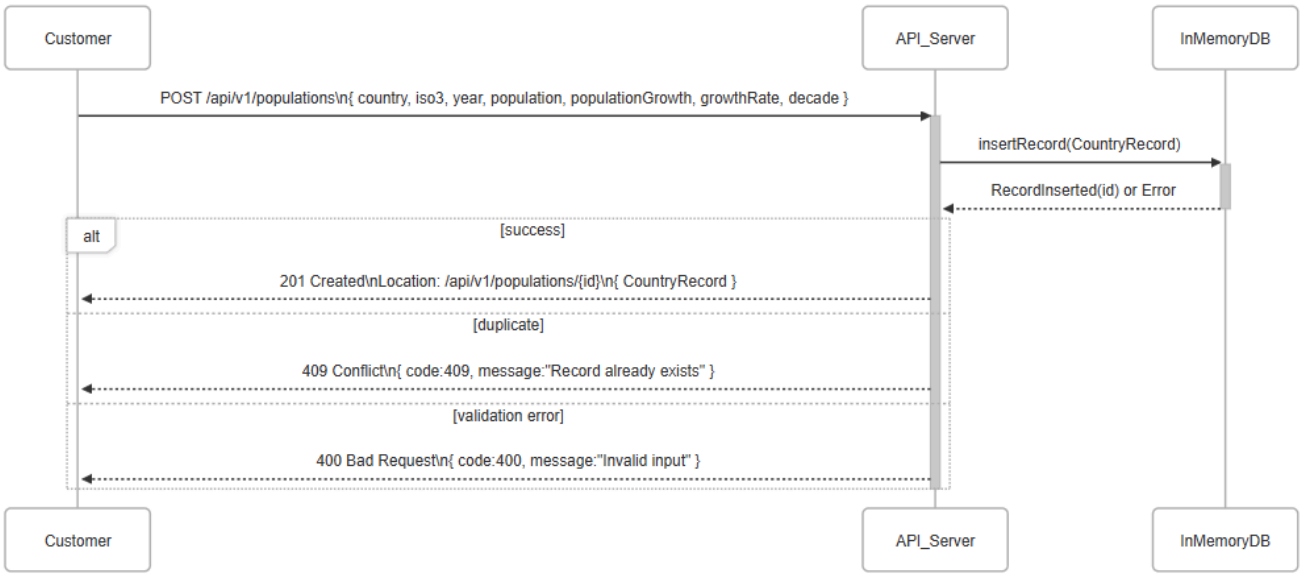
- SharedState is a container that holds a list of CountryRecord entries.
- Each CountryRecord represents one year's population data for a single country.
- A CountryRecord includes fields for the country name, ISO code, year, total population, annual population growth, growth rate percentage, and the decade label.
- The arrow defines a one-to-many relationship i.e. SharedState includes many CountryRecords.

4.4.1 Sequence Diagram 1 - Read Country Record



- The Customer sends a GET request to the API Server asking for the latest country record (by name or ISO3).
- The API Server activates and forwards a query to the In-Memory DB to fetch that country's data.
- The In-Memory DB looks up and returns the matching CountryRecord to the API Server.
- The API Server picks the record with the most recent year and formats it as JSON.
- Finally, the API Server responds to the Customer with a 200 OK status and the requested data.

4.4.2 Sequence Diagram 2 - Create Country Record

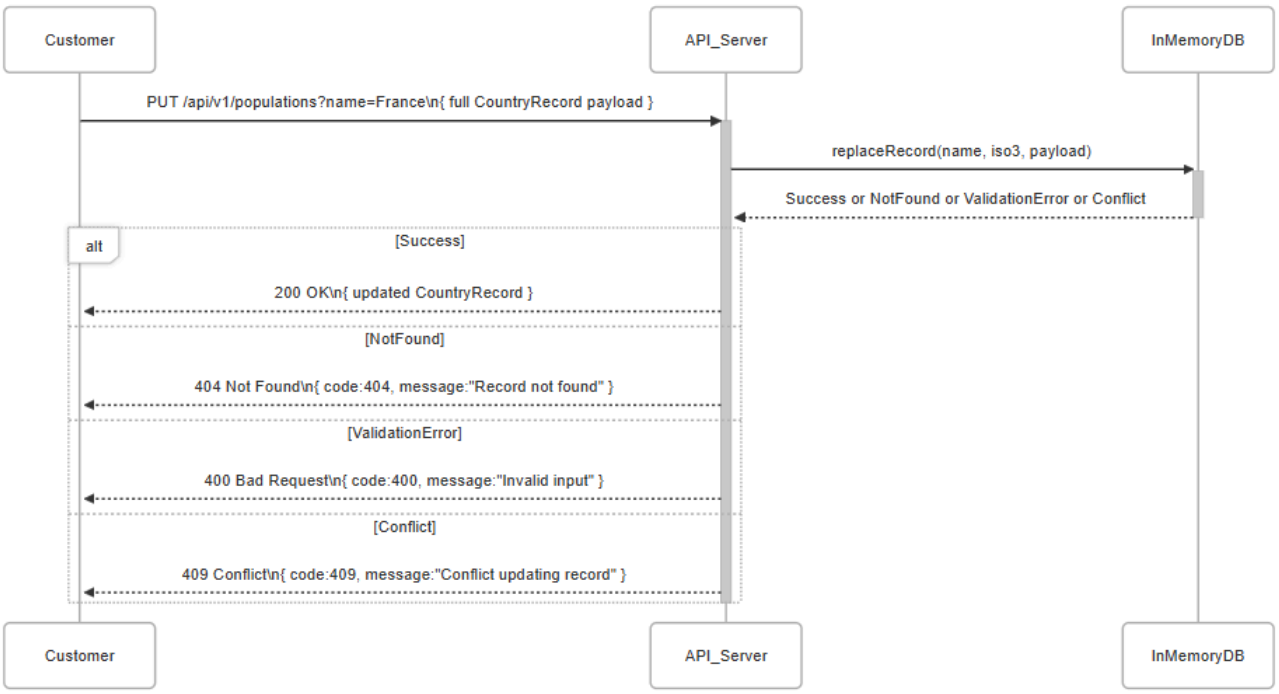


- The Customer sends a POST request with a full country record payload to the API Server.

- The API Server activates and calls the In-Memory DB to insert the new record.
- The In-Memory DB attempts the insert and returns either success or an error (e.g., duplicate key).
- If insertion succeeds, the API Server responds with 201 Created, including the new record (and Location header).
- If the record already exists, the API Server returns 409 Conflict with an error message.
- If the payload fails validation, the API Server returns 400 Bad Request with details.

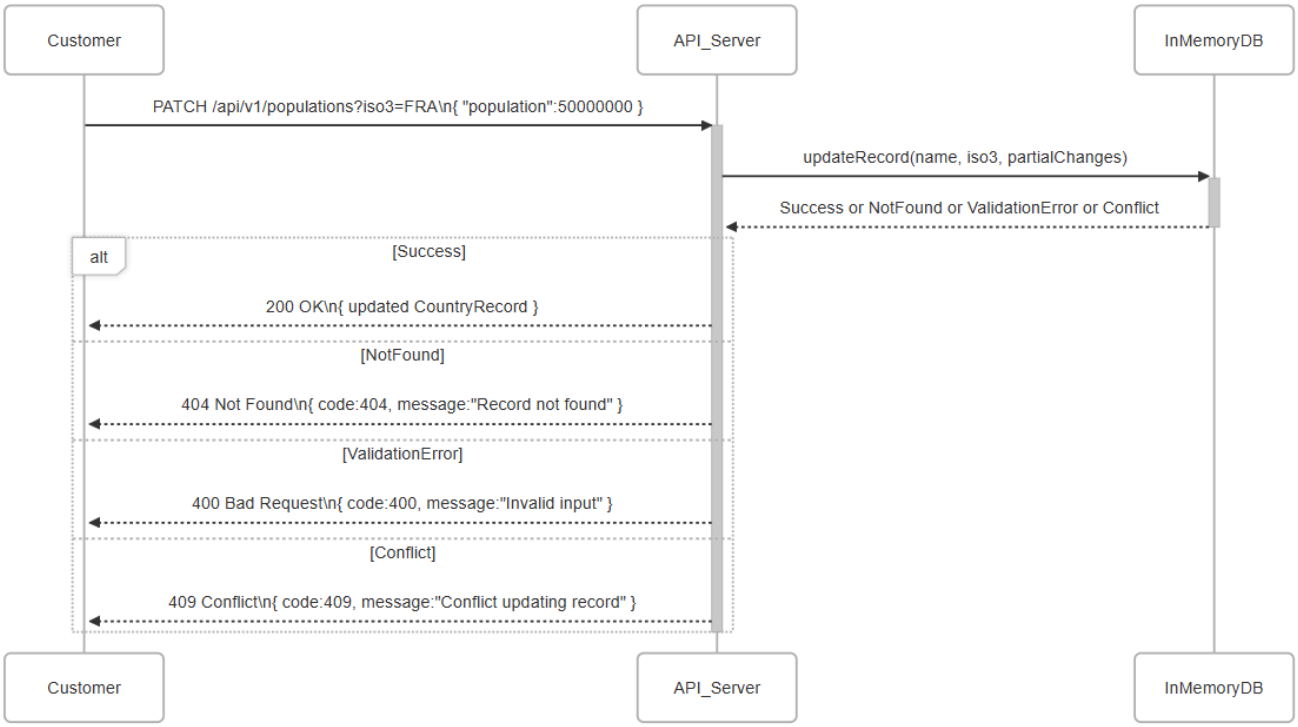
4.4.3 Sequence Diagram 3 - Update Country Record

Update via PUT



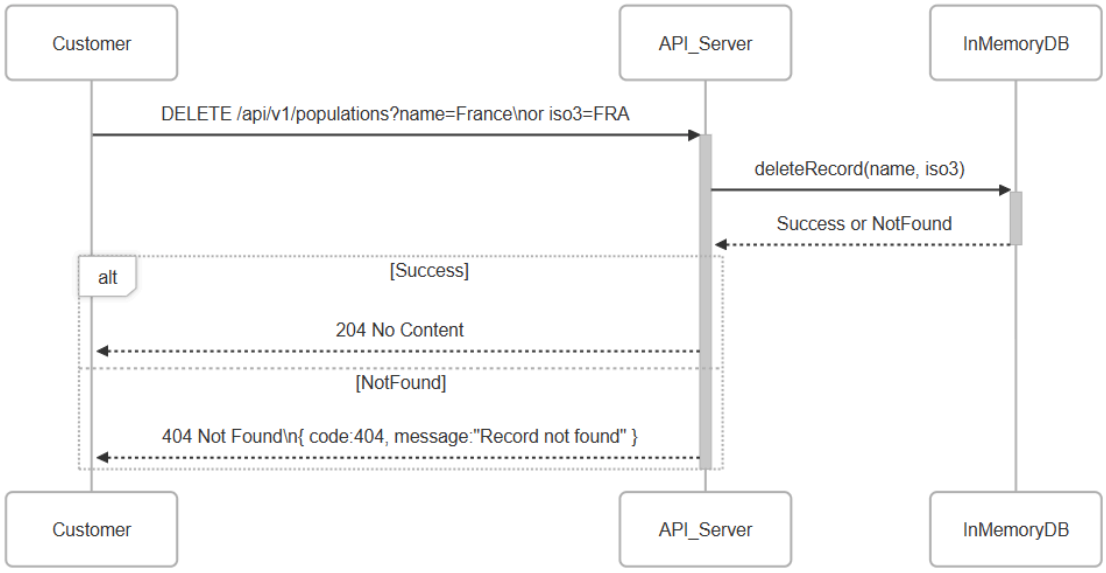
- The Customer sends a PUT request with the complete new country record to the API Server, specifying the country via query (name or ISO3).
- The API Server activates, forwards the replace operation to the In-Memory DB, and waits for the result.
- The In-Memory DB attempts to overwrite the existing record and returns one of: success, not found, validation error, or conflict.
- The API Server then responds to the Customer with:
    1. 200 OK and the updated record on success.
    2. 404 Not Found if no matching record exists.
    3. 400 Bad Request for invalid input.
    4. 409 Conflict if replacing would duplicate or otherwise conflict.

Update via PATCH



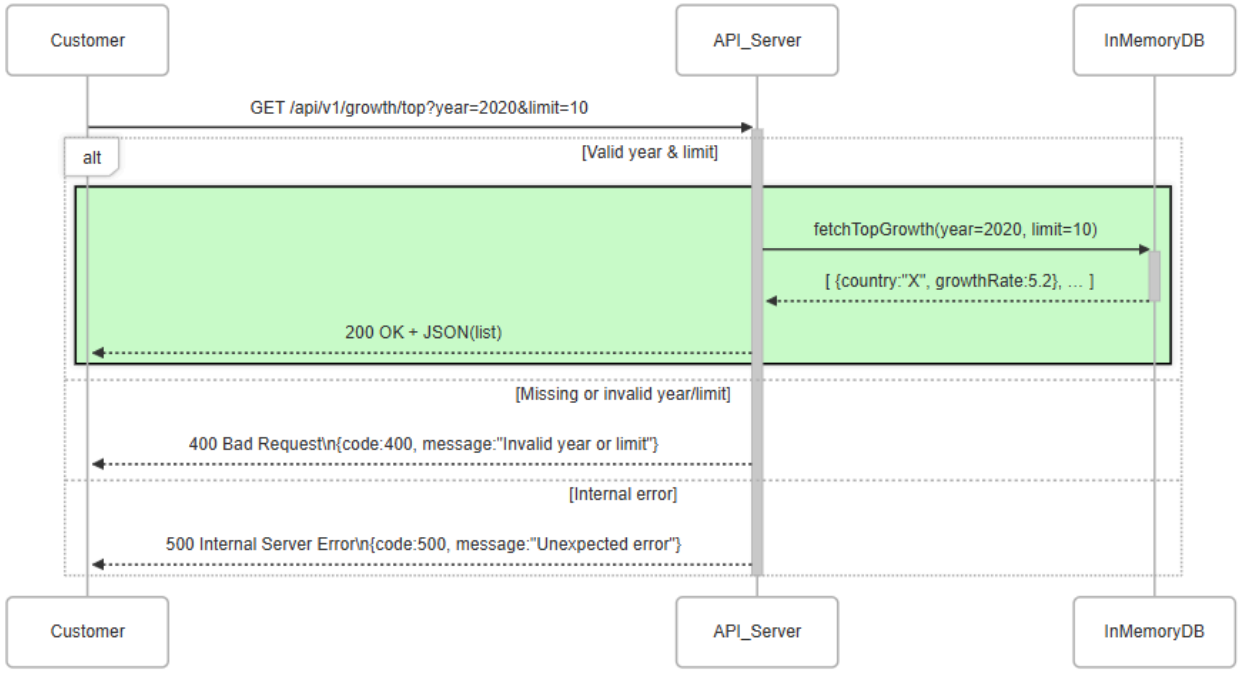
- The Customer sends a PATCH request with only the fields to change (e.g. new population) to the API Server, again identifying the record by name or ISO3.
- The API Server activates and calls the In-Memory DB to apply just those partial updates.
- The In-Memory DB processes the patch and returns success, not found, validation error, or conflict.
- The API Server then replies with:
    1. 200 OK and the updated record on success
    2. 404 Not Found if the record doesn't exist
    3. 400 Bad Request for invalid changes
    4. 409 Conflict if the update would create a duplicate or another conflict

4.4.4 Sequence Diagram 4 - Delete Country Record



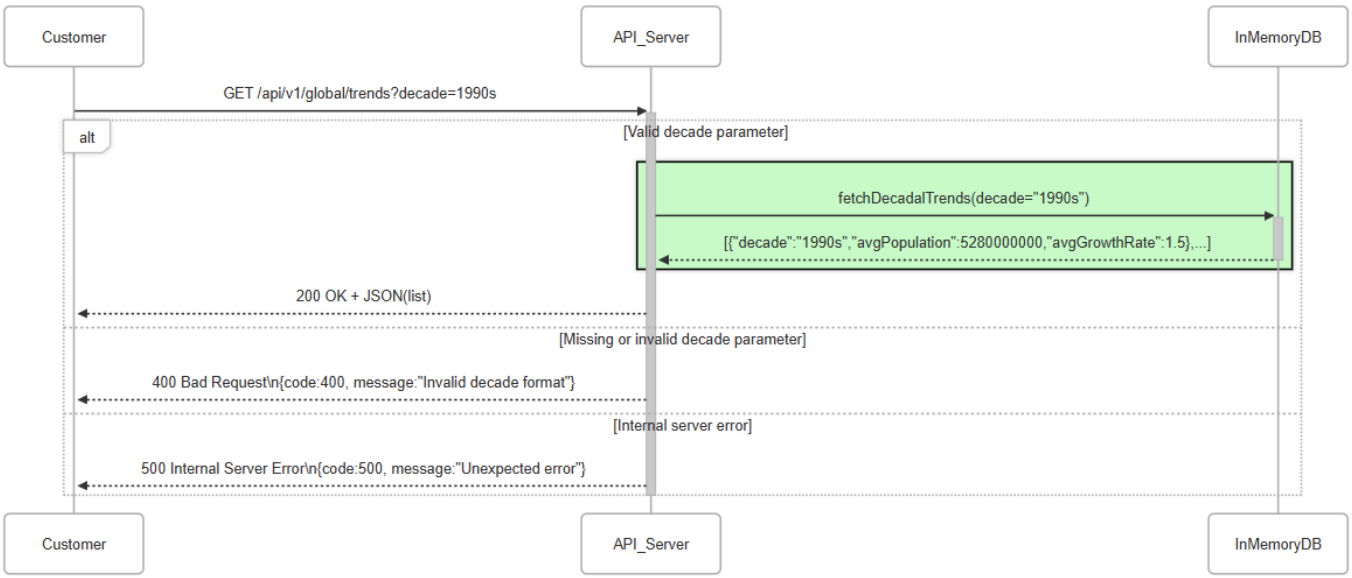
- The Customer sends a DELETE request to the API Server, specifying the country by name or ISO3.
- The API Server activates and calls the In-Memory DB to remove that record.
- The In-Memory DB attempts the deletion and returns either success or “not found.”
- If the record was deleted, the API Server responds with 204 No Content.
- If no matching record exists, the API Server responds with 404 Not Found and an error message.

4.4.5 Sequence Diagram 5 - Fastest Growing Countries



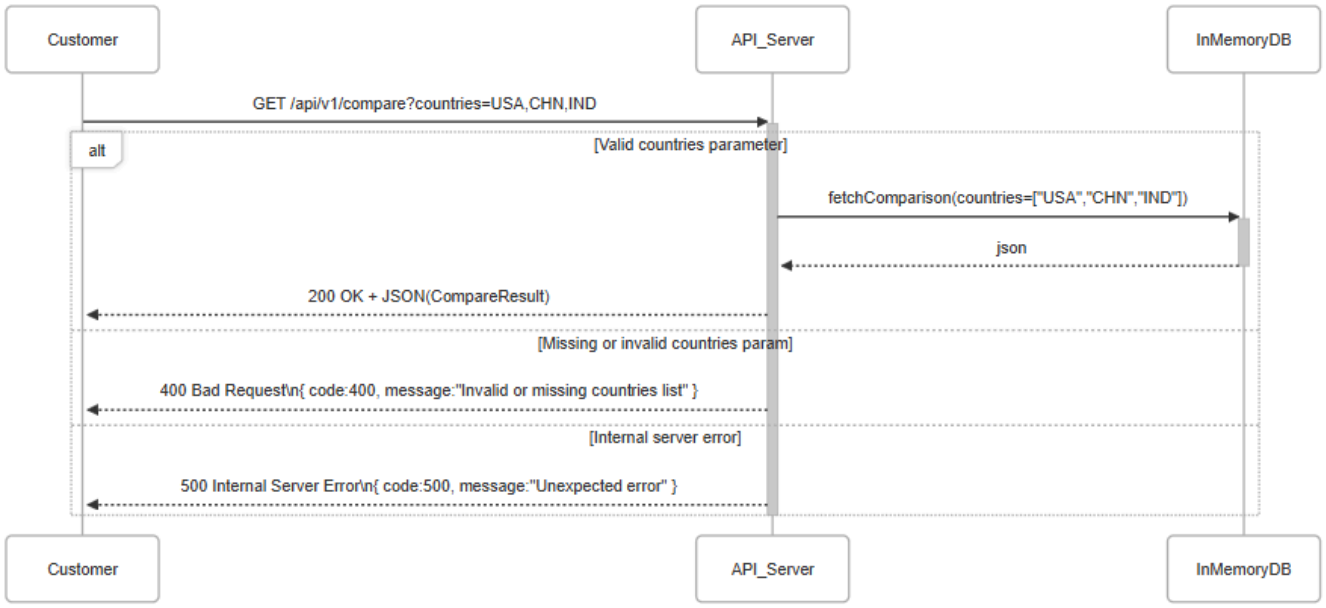
- The Customer sends a GET request for the top-growth countries in a given year with an optional limit.
- If the year and limit parameters are valid, the API Server calls the In-Memory DB to fetch the ranked list.
- The In-Memory DB returns the data, and the API Server responds with 200 OK and the JSON payload.
- If the query parameters are missing or malformed, the API Server immediately returns 400 Bad Request.
- If an unexpected error occurs on the server side, it returns 500 Internal Server Error.

4.4.6 Sequence Diagram 6 - Population Trends Over Decades



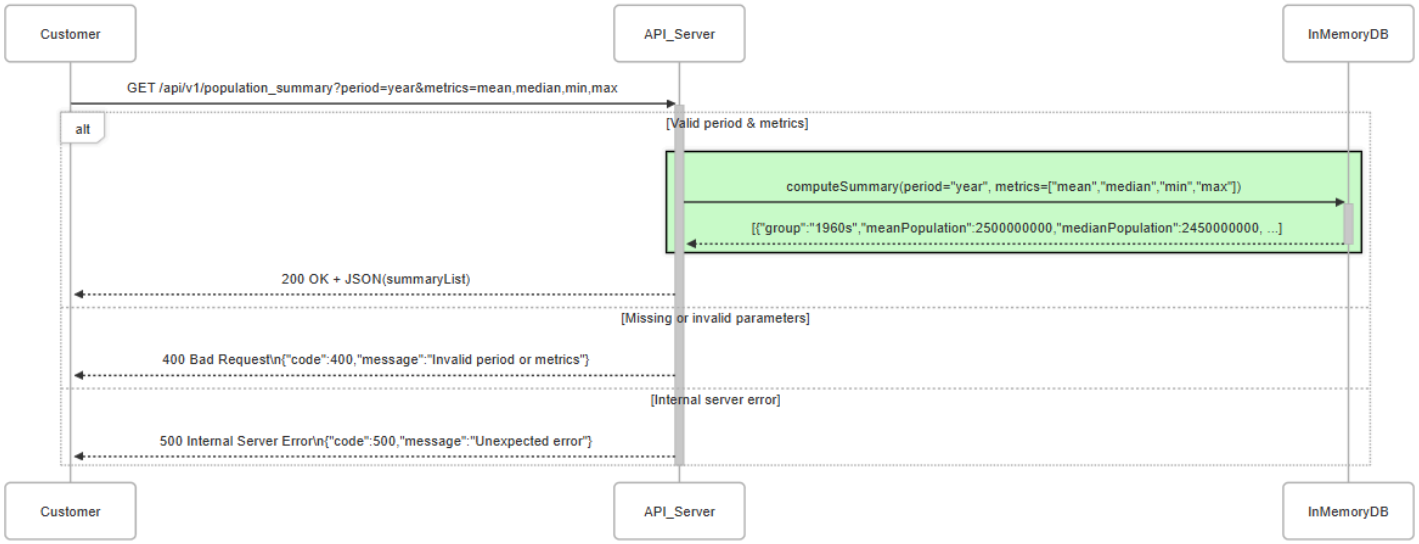
- The Customer sends a GET request to the API Server requesting decadal trends (optionally filtered to a specific decade).
- The API Server activates and calls the In-Memory DB to calculate average population and growth rate by decade.
- The In-Memory DB returns a list of trend objects (each with a decade label, average population, and average growth rate).
- The API Server then responds to the Customer with 200 OK and the JSON payload containing the requested data.

4.4.7 Sequence Diagram 7 - Compare Multi-Country Demographics



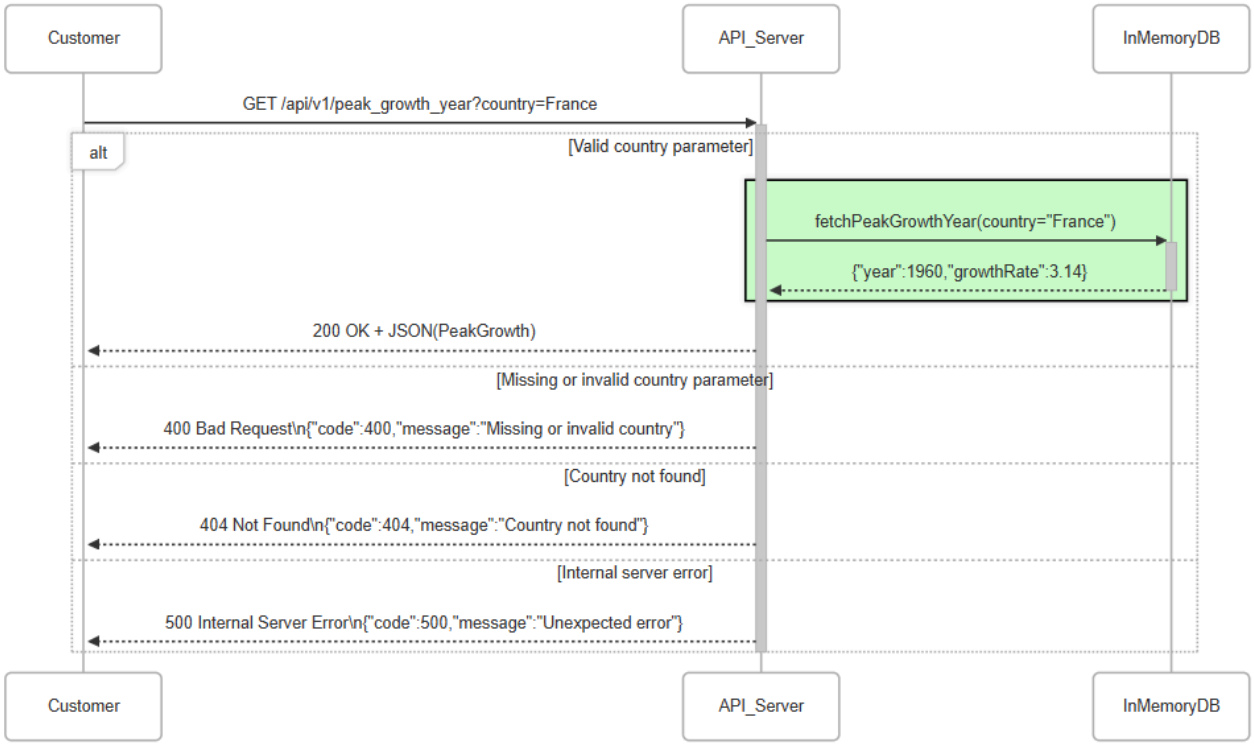
- The Customer sends a GET request to the API Server listing the countries to compare.
- The API Server activates and asks the In-Memory DB to fetch side-by-side time series for each country.
- The In-Memory DB returns a map of country codes to their yearly population and growth-rate lists.
- The API Server packages this data into a CompareResult JSON object and responds with 200 OK.
- The Customer receives the comparison data and can render it side-by-side in their dashboard.

4.4.8 Sequence Diagram 8 - Population Summary Metrics



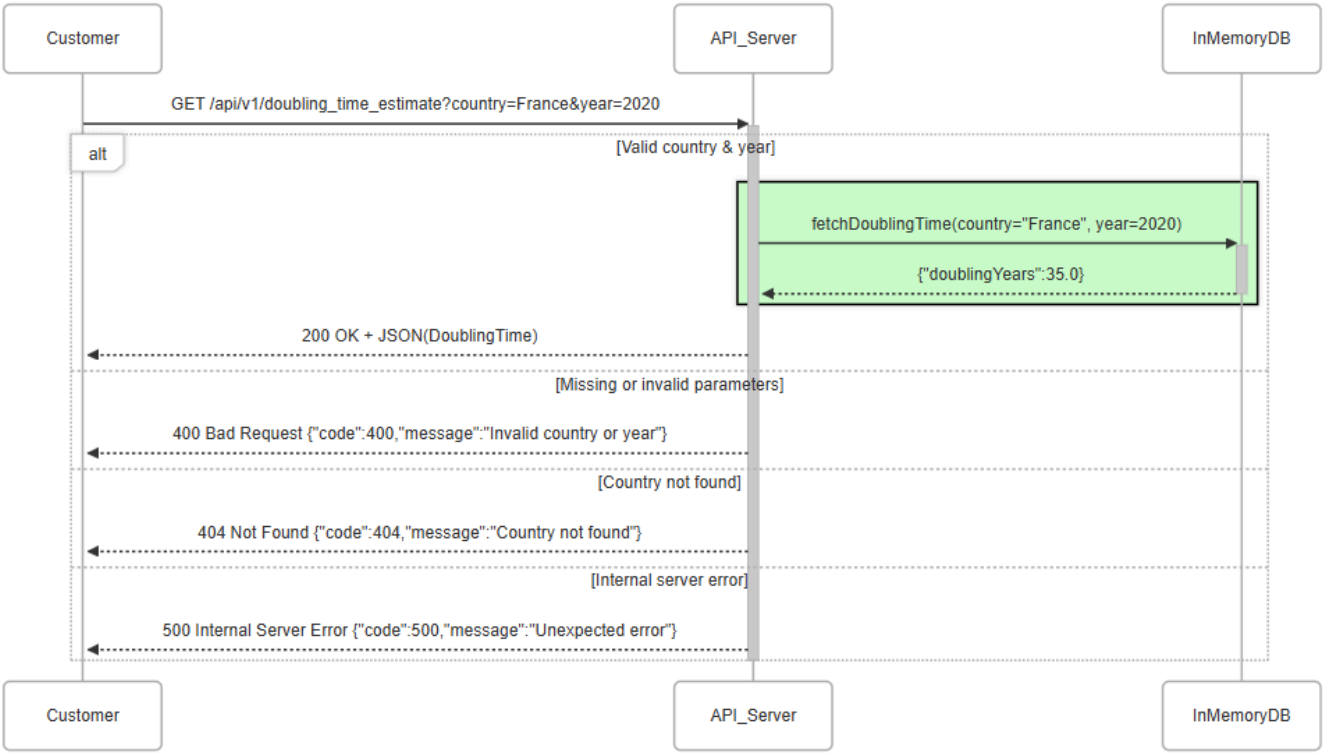
- The Customer sends a GET request to the API Server asking for population summary metrics (mean, median, min, max) grouped by year.
- The API Server checks the period and metrics parameters.
- If valid, the server calls the InMemoryDB to compute the requested summaries, then closes the database lifeline once the data returns.
- The server responds with 200 OK and the JSON list of summary entries, then closes its lifeline.
- If the parameters are invalid, the server immediately returns 400 Bad Request with an error message.
- If something unexpected fails, the server returns 500 Internal Server Error with a generic error message.

4.4.9 Sequence Diagram 9 - Peak Growth Years



- The Customer requests the peak-growth year for a specific country.
- The API Server validates the country parameter.
- If valid, it calls the InMemoryDB to fetch the year and growth-rate peak, then returns 200 OK with that data.
- If the parameter is missing or invalid, it returns 400 Bad Request immediately.
- If no record exists for that country, it returns 404 Not Found.
- On any other failure, it returns 500 Internal Server Error.

4.4.10 Sequence Diagram 10 - Population Doubling Projection



- The Customer requests a doubling-time estimate by sending a GET with country and optional year.
- The API calls the InMemoryDB to calculate the number of years to double.
- The database returns the doublingYears, and the API responds with 200 OK and the JSON result.
- **Missing or invalid parameters:** The API immediately returns 400 Bad Request with an error message.
- **Country not found:** The API returns 404 Not Found if the country isn't in the dataset.
- **Internal server error:**On unexpected failures, the API returns 500 Internal Server Error.

API YMAL

```yaml
openapi: 3.0.3
info:
  title: Population Growth Rate API
  version: 1.0.0
  description: |
    A RESTful API to manage and analyze global population growth data.
servers:
  - url: https://api.example.com

paths:
  /api/v1/population_mappings/country:
    get:
      summary: Read Country Record
      description: Retrieve a country's latest population, growth amount, growth rate, and decade by name or ISO3.
      parameters:
        - in: query
          name: name
          schema:
            type: string
          description: Country name (case-insensitive)
        - in: query
          name: iso3
          schema:
            type: string
          description: Three-letter ISO3 code (case-insensitive)
      responses:
        '200':
          description: Country record found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/CountryRecord'
        '400':
          description: Missing or invalid query parameter
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
        '404':
          description: Country record not found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
        '500':
          description: Internal server error
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
```

```yaml
post:
  summary: Create Country Record
  description: Add a new country population record.
  requestBody:
    required: true
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/CountryRecordInput'
  responses:
    '201':
      description: Record created
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/CountryRecord'
    '400':
      description: Validation error (missing or invalid fields)
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Error'
    '409':
      description: Duplicate record already exists
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Error'
    '500':
      description: Internal server error
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Error'

put:
  summary: Replace Country Record
  description: Fully update an existing country population record by name or ISO3. All fields must be provided.
  parameters:
    - in: query
      name: name
      schema:
        type: string
    - in: query
      name: iso3
      schema:
        type: string
  requestBody:
    required: true
    content:
      application/json:
```

```
        schema:
          $ref: '#/components/schemas/CountryRecordInput'
    responses:
      '200':
        description: Record replaced
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/CountryRecord'
      '400':
        description: Missing or invalid input
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '404':
        description: Country record not found
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '409':
        description: Conflict (would duplicate an existing record)
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '500':
        description: Internal server error
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'

  patch:
    summary: Update Country Record Partially
    description: Update one or more fields of an existing country population record by name or ISO3.
    parameters:
      - in: query
        name: name
        schema:
          type: string
      - in: query
        name: iso3
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/CountryRecordUpdate'
```

```
    responses:
      '200':
        description: Record updated
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/CountryRecord'
      '400':
        description: Missing or invalid input
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '404':
        description: Country record not found
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '409':
        description: Conflict (would duplicate an existing record)
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '500':
        description: Internal server error
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'

  delete:
    summary: Delete Country Record
    description: Remove a country population record by name or ISO3.
    parameters:
      - in: query
        name: name
        schema:
          type: string
      - in: query
        name: iso3
        schema:
          type: string
    responses:
      '204':
        description: Record deleted (no content)
      '400':
        description: Missing or invalid query parameter
        content:
          application/json:
            schema:
```

```
              $ref: '#/components/schemas/Error'
      '404':
        description: Country record not found
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '500':
        description: Internal server error
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'

/api/v1/population_mappings/growth/top:
  get:
    summary: Fastest-Growing Countries
    description: Return the top N countries by population growth rate for a specified year.
    parameters:
      - in: query
        name: year
        required: true
        schema:
          type: integer
          format: int32
      - in: query
        name: limit
        schema:
          type: integer
          format: int32
          default: 10
    responses:
      '200':
        description: List of top-growing countries
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/TopGrowthEntry'
      '400':
        description: Invalid year or limit
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '500':
        description: Internal server error
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
```

```yaml
  /api/v1/population_mappings/global/trends:
    get:
      summary: Decadal Population Trends
      description: Return each decade's average population and growth rate; if a decade is specified, only that
entry is returned.
      parameters:
        - in: query
          name: decade
          schema:
            type: string
            example: "1990s"
      responses:
        '200':
          description: Decadal trends data
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/DecadeTrend'
        '400':
          description: Invalid decade format
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
        '500':
          description: Internal server error
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'

  /api/v1/population_mappings/compare:
    get:
      summary: Compare Multiple Countries' Demographics
      description: Return side-by-side yearly population and growth rate lists for multiple countries.
      parameters:
        - in: query
          name: countries
          required: true
          schema:
            type: array
            items:
              type: string
            example: ["USA","CHN","IND"]
      responses:
        '200':
          description: Comparison data
          content:
            application/json:
```

```
            schema:
              $ref: '#/components/schemas/CompareResult'
      '400':
        description: Missing or invalid countries list
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '500':
        description: Internal server error
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'

/api/v1/population_mappings/population_summary:
  get:
    summary: Population Summary Metrics
    description: Return summary metrics (mean, median, min, max) by year or decade.
    parameters:
      - in: query
        name: period
        required: true
        schema:
          type: string
          enum: [year, decade]
      - in: query
        name: metrics
        required: true
        schema:
          type: array
          items:
            type: string
            enum: [mean, median, min, max]
    responses:
      '200':
        description: Summary metrics
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/SummaryEntry'
      '400':
        description: Invalid period or metrics
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '500':
        description: Internal server error
        content:
```

```
          application/json:
            schema:
              $ref: '#/components/schemas/Error'

  /api/v1/population_mappings/peak_growth_year:
    get:
      summary: Calculate Peak Growth Year
      description: Return the year and value when a country's annual growth rate was highest.
      parameters:
        - in: query
          name: country
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Peak growth year data
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/PeakGrowth'
        '400':
          description: Missing country parameter
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
        '404':
          description: Country not found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'
        '500':
          description: Internal server error
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Error'

  /api/v1/population_mappings/doubling_time_estimate:
    get:
      summary: Population Doubling Projection
      description: Estimate how many years it would take for a country's population to double based on a given
year's growth rate.
      parameters:
        - in: query
          name: country
          required: true
          schema:
            type: string
        - in: query
```

```yaml
        name: year
        schema:
          type: integer
          format: int32
          description: Base year for growth rate (defaults to latest if omitted)
    responses:
      '200':
        description: Doubling time estimate
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/DoublingTime'
      '400':
        description: Missing or invalid parameters
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '404':
        description: Country not found
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '409':
        description: Conflict error (invalid computation scenario)
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'
      '500':
        description: Internal server error
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Error'

components:
  schemas:
    CountryRecord:
      type: object
      properties:
        country:
          type: string
        iso3:
          type: string
        year:
          type: integer
          format: int32
        population:
          type: integer
          format: int64
```

```
      population_growth:
        type: integer
        format: int64
      growth_rate:
        type: number
        format: float
      decade:
        type: string
    required:
      - country
      - iso3
      - year
      - population
      - population_growth
      - growth_rate
      - decade

  CountryRecordInput:
    allOf:
      - $ref: '#/components/schemas/CountryRecord'

  CountryRecordUpdate:
    type: object
    properties:
      population:
        type: integer
        format: int64
      growth_rate:
        type: number
        format: float

  TopGrowthEntry:
    type: object
    properties:
      country:
        type: string
      iso3:
        type: string
      year:
        type: integer
        format: int32
      growth_rate:
        type: number
        format: float
    required:
      - country
      - iso3
      - year
      - growth_rate

  DecadeTrend:
    type: object
```

```
    properties:
      decade:
        type: string
      avg_population:
        type: number
      avg_growth_rate:
        type: number
    required:
      - decade
      - avg_population
      - avg_growth_rate

  CompareResult:
    type: object
    properties:
      countries:
        type: object
        additionalProperties:
          type: array
          items:
            type: object
            properties:
              year:
                type: integer
                format: int32
              population:
                type: integer
                format: int64
              growth_rate:
                type: number
                format: float

  SummaryEntry:
    type: object
    properties:
      group:
        type: string
      mean_population:
        type: number
      median_population:
        type: number
      min_population:
        type: number
      max_population:
        type: number
    required:
      - group

  PeakGrowth:
    type: object
    properties:
      country:
```

```
        type: string
      year:
        type: integer
        format: int32
      growth_rate:
        type: number
        format: float
    required:
      - country
      - year
      - growth_rate

  DoublingTime:
    type: object
    properties:
      country:
        type: string
      base_year:
        type: integer
        format: int32
      doubling_years:
        type: number
    required:
      - country
      - doubling_years

  Error:
    type: object
    properties:
      code:
        type: integer
      message:
        type: string
    required:
      - code
      - message
```

Functional Requirements

5.1 PGR-1 - Country CRUD Endpoints

- **Read** - Returns the country's latest year, population, growth amount, growth rate, and decade.
- **Create** - Adds a new country population record.
- **Delete** - Verify existence and remove country population record.
- **Update** - Updates fields based on name or iso3 and returns the new record.

5.2 PGR-2 - Return Fastest Growing Countries

- Returns the top N countries by population growth rate for that year.

5.3 PGR-3 - Global Population Trends By Decade

- returns each decade's average population and average growth rate.
- If decade is specified,then only that entry is returned.

5.4 PGR-4 - Comparing N Countries

- Returns side-by-side lists of yearly population and growth rate for each country.

5.5 PGR-5: Population Summary Metrics

- Returns the average or median population statistics based on the year or decade.

5.6 PGR-6: Peak Growth Years

- Returns the year in which a specified country experienced its highest annual population growth rate, along with the value of that rate.

5.7 PGR-7: Population Doubling Projection

- Estimates how many years it would take for a country's population to double, based on its growth rate in a given year.

NON-FUNCTIONAL REQUIREMENTS

6.2 Scalability

- The service should support horizontal scaling behind a load-balancer (stateless API servers).
- In-memory data structures should be shardable or cache-backed to handle large datasets without blocking requests.
- Design for eventual migration of the CSV source to a distributed database (e.g., PostgreSQL cluster) with minimal code changes.

6.3 Maintainability

- Follow 12-factor app principles: externalize config, treat logs as event streams, and strictly separate build/runtime.
- Write comprehensive API tests (unit, integration, and contract tests) and enforce code formatting/linting via CI.

6.4 Portability

- The Rust/Actix-Web server must compile and run on Linux, macOS, and Windows with minimal dependencies.
- The front-end (React + Tailwind + Nivo) must work across modern desktop and mobile browsers (Chrome, Safari, Firefox, Edge).
- Dockerize both API and front-end for consistent deployment across environments.

6.5 Availability

- Implement health-check endpoints (GET /healthz) that can be scraped by Kubernetes or any load-balancer.
- Configure automatic restarts on failure via Kubernetes liveness probes.