**Подключение symfony и создание web-api**

1. Настройка окружения (docker-контейнера)

Пример со связкой php+symfony, mysql, nginx-server

* 1. Настройка образа dockerfile на основе alpine

# Используем официальный образ PHP версии 8.2 с FPM (FastCGI Process Manager)

FROM php:8.2-fpm

# Обновляем список пакетов и устанавливаем необходимые пакеты для работы с PHP и его расширениями

RUN apt-get update && apt-get install -y \

wget \

unzip \

curl \

git \

libpng-dev \

libjpeg-dev \

libfreetype6-dev \

libzip-dev \

&& docker-php-ext-install pdo\_mysql mysqli zip

# Устанавливаем расширение OPcache для PHP, которое улучшает производительность за счет кэширования байт-кода

RUN docker-php-ext-install opcache

# Устанавливаем Composer, менеджер зависимостей для PHP, в директорию /usr/local/bin, чтобы он был доступен в PATH

RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer

# Устанавливаем Symfony CLI, инструмент командной строки для работы с Symfony 2й

RUN wget https://get.symfony.com/cli/installer -O - | bash && \

    mv /root/.symfony5/bin/symfony /usr/local/bin/symfony

# Устанавливаем Symfony Maker Bundle, который предоставляет команды для генерации кода в Symfony

RUN composer require symfony/maker-bundle --dev

# Устанавливаем пакет Symfony Serializer Pack, который включает в себя сериализатор и другие полезные компоненты

RUN composer require symfony/serializer-pack --dev

# Устанавливаем компонент Symfony Serializer, который предоставляет функциональность для сериализации и десериализации объектов

RUN composer require symfony/serializer --dev

# Фиксация совместимых версий

RUN composer require phpstan/phpdoc-parser:^1.8 symfony/property-info:^6.3

# Открываем порт 9000 для доступа к PHP-FPM извне контейнера

EXPOSE 9000

* 1. Настройка docker-compose.yml

services:

  # Сервис приложения

  app:

    build:

      context: .  # Собирает образ из Dockerfile в текущей директории

    container\_name: world-countries-app  # Имя контейнера

    volumes:

      - ./app:/var/www/app  # Монтирует папку ./app в контейнер по пути /var/www/app

    working\_dir: /var/www/app  # Рабочая директория внутри контейнера

    environment:

      - DB\_HOST=${DB\_HOST}  # Хост БД (берётся из переменных окружения)

      - DB\_PORT=${DB\_PORT}  # Порт БД

      - DB\_USERNAME=${DB\_USERNAME}  # Пользователь БД

      - DB\_PASSWORD=${DB\_PASSWORD}  # Пароль БД

      - DB\_NAME=${DB\_NAME}  # Имя БД

    ports:

      - "9000:9000"  # Пробрасывает порт 9000 контейнера на порт 9000 хоста

  # Сервис базы данных MySQL

  db:

    container\_name: world-countries-db  # Имя контейнера

    image: mysql:latest  # Использует официальный образ MySQL

    ports:

      - "3307:3306"  # Пробрасывает порт 3306 контейнера на порт 3307 хоста

    environment:

      - MYSQL\_ROOT\_PASSWORD={DB\_PASSWORD}  # Пароль root-пользователя (фигурные скобки - вероятно опечатка, должны быть ${DB\_PASSWORD})

      - MYSQL\_DATABASE=${DB\_NAME}  # Создаёт БД при старте

      - MYSQL\_USER=${USER\_NAME}

      - MYSQL\_PASSWORD=${USER\_PASSWORD}

    volumes:

      - ./mysql/init:/docker-entrypoint-initdb.d  # Монтирует SQL-скрипты для инициализации БД

  # Сервис веб-сервера Nginx

  nginx:

    image: nginx:latest  # Использует официальный образ Nginx

    container\_name: world-countries-server  # Имя контейнера

    volumes:

      - ./app:/var/www/app  # Монтирует папку с приложением

      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf  # Монтирует конфиг Nginx

    ports:

      - "8080:80"  # Пробрасывает порт 80 контейнера на порт 8080 хоста

    depends\_on:

      - app  # Запускается только после запуска сервиса app

1.3. Настройка файла с переменными окружения .env

# параметры подключения к БД

DB\_HOST="db"

DB\_PORT=3306

DB\_USERNAME="root"

DB\_PASSWORD="root"

DB\_NAME="world-countries\_db"

USER\_NAME="vi"

USER\_PASSWORD="vi"

1.4. Предварительная инициализация БД

worldcountriesweb-app\mysql\init\init.sql

-- удаление таблицы, если она уже существует

DROP TABLE IF EXISTS countries\_t;

-- установка кодировки соединения и базы

SET NAMES 'utf8mb4';

SET CHARACTER SET utf8mb4;

SET collation\_connection = 'utf8mb4\_unicode\_ci';

--  создание таблиц стран

-- shortName VARCHAR(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

CREATE TABLE countries\_t (

    id INT NOT NULL AUTO\_INCREMENT,

    shortName VARCHAR(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

    fullName VARCHAR(100) CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci NOT NULL,

    isoAlpha2 CHAR(2) NOT NULL,

    isoAlpha3 CHAR(3) NOT NULL,

    isoNumeric CHAR(3) NOT NULL,

    countryPopulation BIGINT NOT NULL,

    countrySquare FLOAT NOT NULL,

    PRIMARY KEY(id),

    UNIQUE(isoAlpha2),

    UNIQUE(isoAlpha3),

    UNIQUE(isoNumeric)

);

-- добал=вление сетовых данных

INSERT INTO countries\_t (

    shortName,

    fullName,

    isoAlpha2,

    isoAlpha3,

    isoNumeric,

    countryPopulation,

    countrySquare

) VALUES

('Россия', 'Российская Федерация', 'RU', 'RUS', '643', 146150789, 17125191),

('Беларусь', 'Республика Беларусь', 'BY', 'BLR', '112', 9397800, 207600),

('США', 'Соединённые Штаты Америки', 'US', 'USA', '840', 331893745, 9833517),

('Китай', 'Китайская Народная Республика', 'CN', 'CHN', '156', 1411750000, 9596961),

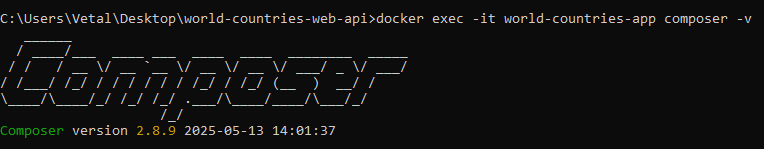
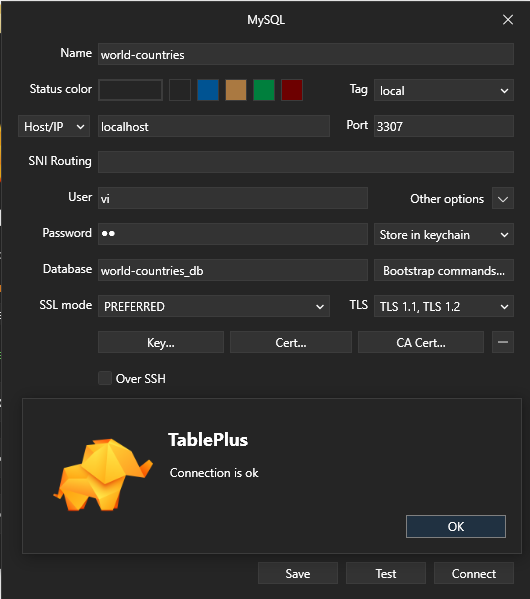
('Андорра', 'Княжество Андорра', 'AD', 'AND', '020', 79824, 0.47),

('Вьетнам', 'Социалистическая Республика Вьетнам', 'VN', 'VNM', '704', 98168829, 331212);

2. Запуск и проверка контейнера

2.1. Скачивание, сборка и запуск: docker compose up –d

2.2. Проверка работоспособности:



2.3. Создание проекта

docker exec -it world-countries-app composer create-project symfony/skeleton:"7.0.\*" /var/www/app

Проект создается в директории app

Пояснение:

1. **docker exec -it** - выполняет команду в запущенном контейнере с интерактивным терминалом
2. **world-countries-app** - имя контейнера
3. **composer create-project** - команда Composer для создания нового проекта
4. **symfony/skeleton:"7.0.\*"** - устанавливает Symfony Skeleton версии 7.0.\* (последнюю стабильную в ветке 7.0)

3. Создание контроллеров

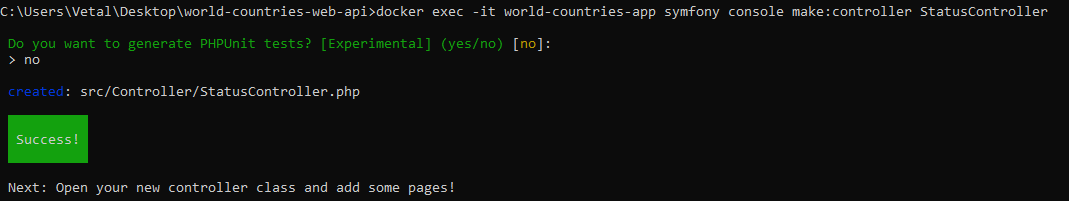
3.1. Добавление генератора кода (бандл Symfony) maker-bundle

Для добавления контроллеров нужно установить maker-bundle

docker exec -it world-countries-app composer require symfony/maker-bundle --dev

3.2. Создание тестового контроллера StatusController

docker exec -it world-countries-app symfony console make:controller StatusController



3.3. Редактирование контроллеров

<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\HttpFoundation\JsonResponse;

use Symfony\Component\Routing\Attribute\Route;

use Symfony\Component\HttpFoundation\Request;

final class StatusController extends AbstractController

{

    #[Route('/', name: 'app\_status' , methods: ['GET'])]

    public function index(Request $request): JsonResponse

    {

        $data = [

        'status' => 'server is running',

        'host' => $request->getHost(),

        'protocol' => $request->getScheme()

        ];

        return new JsonResponse($data);

    }

    #[Route('/ping', name: 'api\_ping', methods: ['GET'])]

    public function ping(): JsonResponse

    {

        return $this->json([

            'message' => 'pong'

        ]);

    }

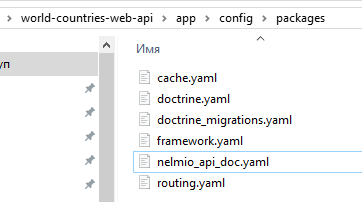
}

3.4. Добавление swagger

* Установка необходимых пакетов:

docker exec -it world-countries-app composer require symfony/orm-pack symfony/maker-bundle nelmio/api-doc-bundle: doctrine/doctrine-fixtures-bundle

В вопросах ответить (yes)



Должны появиться следующие файлы

* Настройка конечной точки контроллеров (#[OA\Tag] помечает конечную точку для пользовательского интерфейса Swagger. )

<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\HttpFoundation\JsonResponse;

use Symfony\Component\Routing\Attribute\Route;

use Symfony\Component\HttpFoundation\Request;

use OpenApi\Attributes as OA;

final class StatusController extends AbstractController

{

    #[Route('/', name: 'app\_status' , methods: ['GET'])]

    #[OA\Tag(name: 'Status')]

    public function index(Request $request): JsonResponse

    {

        $data = [

        'status' => 'server is running',

        'host' => $request->getHost(),

        'protocol' => $request->getScheme()

        ];

        return new JsonResponse($data);

    }

    #[Route('/ping', name: 'api\_ping', methods: ['GET'])]

    #[OA\Tag(name: 'Ping')]

    public function ping(): JsonResponse

    {

        return $this->json([

            'message' => 'pong'

        ]);

    }

}

* Настройка документации OpenAPI (config/packages/nelmio\_api\_doc.yaml)

nelmio\_api\_doc:

    documentation:

        info:

            title: My App

            description: This is an awesome app!

            version: 1.0.0

    areas: # to filter documented areas

        path\_patterns:

            - ^/api(?!/doc$) # Accepts routes under /api except /api/doc

* Файл openapi.yaml

openapi: 3.1.0

info:

  title: Справочник стран мира API

  version: 1.0.0

  description: API для работы с веб-приложенем "Справочник стран мира"

  # contact:

  #   email: support@api.com

  # license:

  #   name: MIT

  #   url: https://opensource.org/licenses/MIT

servers:

  - url: http://localhost:8080

    description: Локальная среда разработки

  # - url: https://api.example.com

  #   description: Продакшен сервер

tags:

  - name: Status

    description: Проверка состояния сервера

  - name: Ping

    description: Проверка доступности API

  - name: Countries

    description: Управление данными стран

paths:

  /:

    get:

      tags:

        - Status

      summary: Проверка статуса сервера

      description: Возвращает текущее состояние сервера

      responses:

        '200':

          description: Успешный ответ

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/StatusResponse'

  /ping:

    get:

      tags:

        - Ping

      summary: Ping endpoint

      description: Простая проверка доступности API

      responses:

        '200':

          description: Успешный ответ

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/PingResponse'

  /api/country:

    get:

      tags:

        - Countries

      summary: Получить список всех стран

      responses:

        '200':

          description: Список стран

          content:

            application/json:

              schema:

                type: array

                items:

                  $ref: '#/components/schemas/Country'

    post:

      tags:

        - Countries

      summary: Добавить новую страну

      requestBody:

        required: true

        content:

          application/json:

            schema:

              $ref: '#/components/schemas/Country'

      responses:

        '200':

          description: Страна успешно добавлена

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Country'

        '400':

          description: Неверный код страны

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Error'

        '409':

          description: Код страны уже существует

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Error'

  /api/country/{code}:

    get:

      tags:

        - Countries

      summary: Получить страну по коду

      parameters:

        - name: code

          in: path

          required: true

          schema:

            type: string

          description: Код страны (ISO Alpha-2)

      responses:

        '200':

          description: Данные страны

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Country'

        '400':

          description: Неверный код страны

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Error'

        '404':

          description: Страна не найдена

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Error'

    patch:

      tags:

        - Countries

      summary: Обновить данные страны

      parameters:

        - name: code

          in: path

          required: true

          schema:

            type: string

          description: Код страны (ISO Alpha-2)

      requestBody:

        required: true

        content:

          application/json:

            schema:

              $ref: '#/components/schemas/Country'

      responses:

        '200':

          description: Данные страны успешно обновлены

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Country'

        '404':

          description: Страна не найдена

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Error'

    delete:

      tags:

        - Countries

      summary: Удалить страну

      parameters:

        - name: code

          in: path

          required: true

          schema:

            type: string

          description: Код страны (ISO Alpha-2)

      responses:

        '204':

          description: Страна успешно удалена

        '404':

          description: Страна не найдена

          content:

            application/json:

              schema:

                $ref: '#/components/schemas/Error'

components:

  schemas:

    StatusResponse:

      type: object

      properties:

        status:

          type: string

          example: "server is running"

          description: Статус сервера

        host:

          type: string

          example: "localhost"

          description: Хост сервера

        protocol:

          type: string

          example: "http"

          description: Используемый протокол

        timestamp:

          type: string

          format: date-time

          example: "2023-05-15T12:00:00+00:00"

          description: Временная метка ответа

      required:

        - status

        - host

        - protocol

        - timestamp

    PingResponse:

      type: object

      properties:

        message:

          type: string

          example: "pong"

          description: Ответное сообщение

        timestamp:

          type: string

          format: date-time

          example: "2023-05-15T12:00:00+00:00"

          description: Временная метка ответа

      required:

        - message

        - timestamp

    Country:

      type: object

      properties:

        shortName:

          type: string

          example: "Dreamland"

          description: Short name of the country

        fullName:

          type: string

          example: "SuperDuper Dreamland"

          description: Full official name of the country

        isoAlpha2:

          type: string

          example: "DL"

          description: alpha-2 country code

          maxLength: 2

          minLength: 2

        isoAlpha3:

          type: string

          example: "SDL"

          description: alpha-3 country code

          maxLength: 3

          minLength: 3

        isoNumeric:

          type: integer

          example: 888

          description: numeric country code

        population:

          type: integer

          example: 100500

          description: Country population

        square:

          type: number

          format: float

          example: 100500

          description: Country area in square kilometers

      required:

        - shortName

        - fullName

        - isoAlpha2

        - isoAlpha3

        - isoNumeric

        - population

        - square

    Error:

      type: object

      properties:

        errorCode:

          type: integer

          example: 404

          description: HTTP status code

        errorMessage:

          type: string

          example: "Country not found"

          description: Error message

        timestamp:

          type: string

          format: date-time

          example: "2023-05-15T12:00:00+00:00"

          description: Временная метка ошибки

      required:

        - errorCode

        - errorMessage

        - timestamp

* Создание OpenAPI YAML

Создать папку с документацией:

docker exec -it world-countries-app mkdir -p public/docs

docker exec -it world-countries-app php bin/console nelmio:apidoc:dump --format=yaml > public/docs/openapi.yaml

(спарсить не получилось, в конфиг openapi.yaml данные добавлял через DeepSeek)

* Установить npm:

docker exec -it world-countries-app apt install npm

* Установка Swagger UI через npm

docker exec -it world-countries-app npm install swagger-ui-dist

* Скопировать папку dist в public/docs:

cp -r node\_modules/swagger-ui-dist/\* public/docs/

* Создать маршруты для /docs и элементов swagger :

docker exec -it world-countries-app symfony console make:controller DocsController

<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\HttpFoundation\JsonResponse;

use Symfony\Component\Routing\Attribute\Route;

use Symfony\Component\HttpFoundation\Response;

final class DocsController extends AbstractController

{

        #[Route('/docs', name: 'api\_docs')]

        public function index(): Response

        {

            return new Response(file\_get\_contents(\_\_DIR\_\_ . '/../../public/docs/index.html'));

        }

        #[Route('/swagger-ui.css', name: 'swagger\_css')]

    public function swaggerCss(): Response

    {

        return new Response(

            file\_get\_contents(\_\_DIR\_\_ . '/../../public/docs/swagger-ui.css'),

            200,

            ['Content-Type' => 'text/css']

        );

    }

    #[Route('/swagger-ui-standalone-preset.js', name: 'swagger\_standalone\_js')]

    public function swaggerStandaloneJs(): Response

    {

        return new Response(

            file\_get\_contents(\_\_DIR\_\_ . '/../../public/docs/swagger-ui-standalone-preset.js'),

            200,

            ['Content-Type' => 'application/javascript']

        );

    }

    #[Route('/swagger-ui-bundle.js', name: 'swagger\_bundle\_js')]

    public function swaggerBundleJs(): Response

    {

        return new Response(

            file\_get\_contents(\_\_DIR\_\_ . '/../../public/docs/swagger-ui-bundle.js'),

            200,

            ['Content-Type' => 'application/javascript']

        );

    }

    #[Route('/index.css', name: 'index\_css')]

    public function indexCss(): Response

    {

        return new Response(

            file\_get\_contents(\_\_DIR\_\_ . '/../../public/docs/index.css'),

            200,

            ['Content-Type' => 'text/css']

        );

    }

    #[Route('/swagger-initializer.js', name: 'swagger\_initializer')]

    public function swaggerInitializer(): Response

    {

        return new Response(

            file\_get\_contents(\_\_DIR\_\_ . '/../../public/docs/swagger-initializer.js'),

            200,

            ['Content-Type' => 'application/javascript']

        );

    }

    #[Route('/favicon-32x32.png', name: 'favicon\_32')]

    public function favicon32(): Response

    {

        return new Response(

            file\_get\_contents(\_\_DIR\_\_ . '/../../public/docs/favicon-32x32.png'),

            200,

            ['Content-Type' => 'image/png']

        );

    }

    #[Route('/favicon-16x16.png', name: 'favicon\_16')]

    public function favicon16(): Response

    {

        return new Response(

            file\_get\_contents(\_\_DIR\_\_ . '/../../public/docs/favicon-16x16.png'),

            200,

            ['Content-Type' => 'image/png']

        );

    }

}

* Модифицировать файл swagger-initializer.js следующим образом:

Осдержимое файла:

const ui = SwaggerUIBundle({

url: "/docs/openapi.yaml",

dom\_id: '#swagger-ui',

presets: [

SwaggerUIBundle.presets.apis,

SwaggerUIStandalonePreset

],

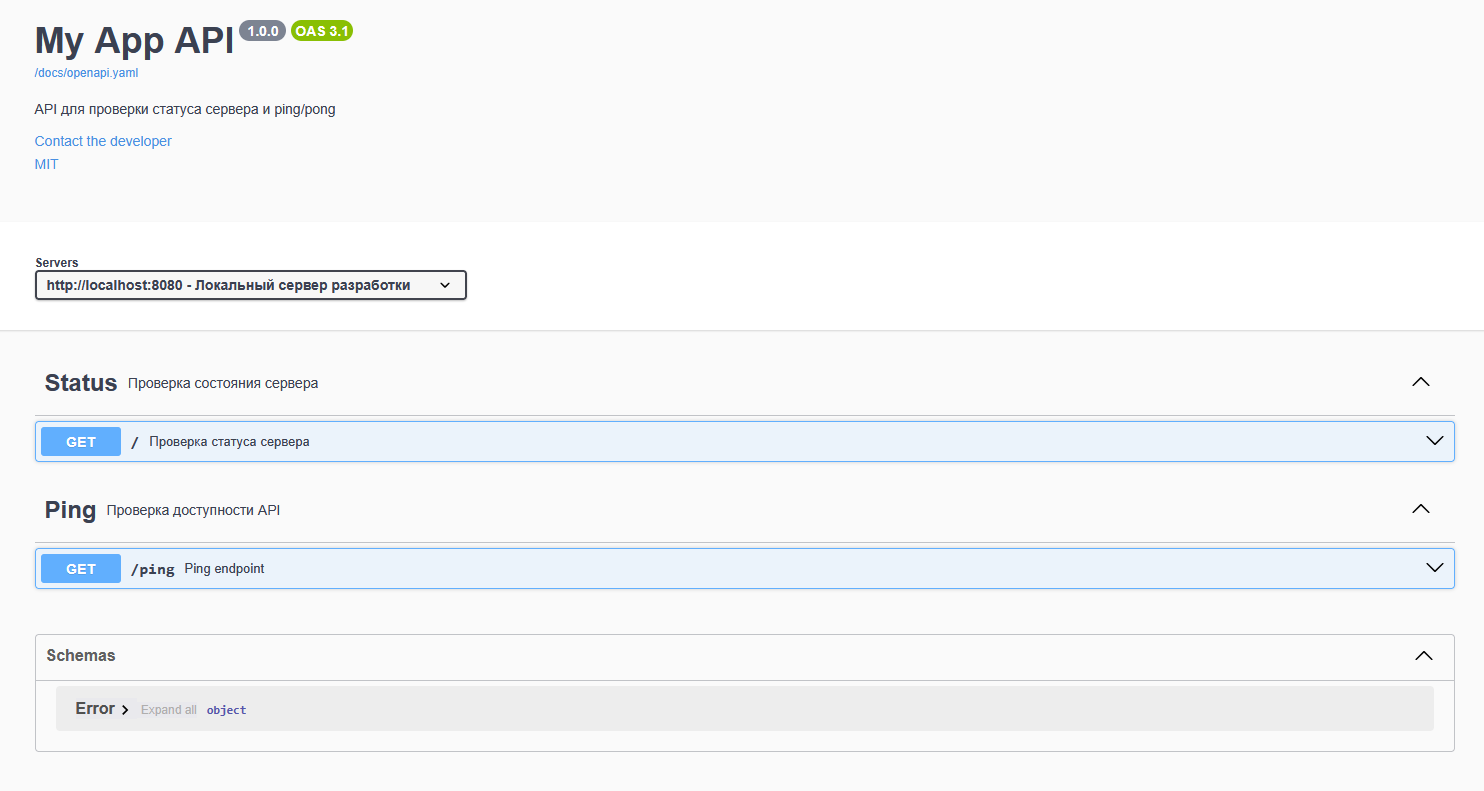
layout: "StandaloneLayout"

});

* Очистить кэш

php bin/console cache:clear

При запросе на <http://localhost:8080/docs> должно появиться следующее окно



Статья исходник:

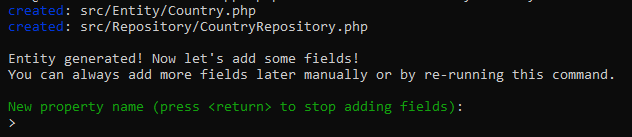
https://brainstreamtechnolabs.com/build-symfony-rest-api-with-openapi-and-swagger-ui-integration/

4. Создание моделей

docker exec -it world-countries-app bash

php bin/console make:entity Country

Команда php bin/console make:entity в Symfony — это интерактивный инструмент для создания сущностей Doctrine ORM (классов PHP, которые представляют таблицы в базе данных). Вот что она делает:



$ php bin/console make:entity

Class name of the entity to create or update:

> Product

New property name (press <return> to stop adding fields):

> name

Field type [string]:

> string

Field length [255]:

> 100

Can this field be null in the database (nullable) (yes/no) [no]:

> no

New property name:

> price

Field type [string]:

> float

<?php

namespace App\Entity;

use App\Repository\CountryRepository;

use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity(repositoryClass: CountryRepository::class)]

class Country

{

    #[ORM\Id]

    #[ORM\GeneratedValue]

    #[ORM\Column]

    private ?int $id = null;

    #[ORM\Column(length: 255)]

    private ?string $shortName = null;

    #[ORM\Column(length: 255)]

    private ?string $fullName = null;

    #[ORM\Column(length: 2)]

    private ?string $isoAlpha2 = null;

    #[ORM\Column(length: 3)]

    private ?string $isoAlpha3 = null;

    #[ORM\Column(length: 255)]

    private ?string $isoNumeric = null;

    #[ORM\Column]

    private ?int $population = null;

    #[ORM\Column]

    private ?float $square = null;

    public function getId(): ?int

    {

        return $this->id;

    }

    public function getShortName(): ?string

    {

        return $this->shortName;

    }

    public function setShortName(string $shortName): static

    {

        $this->shortName = $shortName;

        return $this;

    }

    public function getFullName(): ?string

    {

        return $this->fullName;

    }

    public function setFullName(string $fullName): static

    {

        $this->fullName = $fullName;

        return $this;

    }

    public function getIsoAlpha2(): ?string

    {

        return $this->isoAlpha2;

    }

    public function setIsoAlpha2(string $isoAlpha2): static

    {

        $this->isoAlpha2 = $isoAlpha2;

        return $this;

    }

    public function getIsoAlpha3(): ?string

    {

        return $this->isoAlpha3;

    }

    public function setIsoAlpha3(string $isoAlpha3): static

    {

        $this->isoAlpha3 = $isoAlpha3;

        return $this;

    }

    public function getIsoNumeric(): ?string

    {

        return $this->isoNumeric;

    }

    public function setIsoNumeric(string $isoNumeric): static

    {

        $this->isoNumeric = $isoNumeric;

        return $this;

    }

    public function getPopulation(): ?int

    {

        return $this->population;

    }

    public function setPopulation(int $population): static

    {

        $this->population = $population;

        return $this;

    }

    public function getSquare(): ?float

    {

        return $this->square;

    }

    public function setSquare(float $square): static

    {

        $this->square = $square;

        return $this;

    }

}

Данный метод – делает правильно, но сложно. Для упрощения модели выглядят седующим образом:

<?php

namespace App\Entity;

use OpenApi\Attributes as OA;

#[OA\Schema(

    schema: 'Country',

    title: 'Country',

    description: 'Country entity',

    required: ['shortName', 'fullName', 'isoAlpha2', 'isoAlpha3', 'isoNumeric', 'population', 'square']

)]

class Country

{

    public function \_\_construct(

        #[OA\Property(

            type: 'string',

            example: 'ExampleCountry',

            description: 'Short name of the country'

        )]

        public string $shortName,

        #[OA\Property(

            type: 'string',

            example: 'Dreamland',

            description: 'SuperDuper Dreamland'

        )]

        public string $fullName,

        #[OA\Property(

            type: 'string',

            example: 'DL',

            description: 'alpha-2 country code',

            maxLength: 2,

            minLength: 2

        )]

        public string $isoAlpha2,

        #[OA\Property(

            type: 'string',

            example: 'SDL',

            description: 'alpha-3 country code',

            maxLength: 3,

            minLength: 3

        )]

        public string $isoAlpha3,

        #[OA\Property(

            type: 'integer',

            example: 888,

            description: 'numeric country code'

        )]

        public int $isoNumeric,

        #[OA\Property(

            type: 'integer',

            example: 146599183,

            description: 'Country population'

        )]

        public int $population,

        #[OA\Property(

            type: 'number',

            format: 'float',

            example: 17098242.0,

            description: 'Country area in square kilometers'

        )]

        public float $square

    ) {

    }

}

#[OA\Schema(

    schema: 'Error',

    title: 'Error',

    description: 'Error response',

    required: ['errorCode', 'errorMessage']

)]

class Error

{

    #[OA\Property(

        type: 'integer',

        example: 404,

        description: 'HTTP status code'

    )]

    public int $errorCode;

    #[OA\Property(

        type: 'string',

        example: 'Country not found',

        description: 'Error message'

    )]

    public string $errorMessage;

}

5. Создание сценариев работы и исключений

/app/src/Entity/CountryScennarios.php

<?php

namespace App\Entity;

use App\Entity\Countrystoragesitory;

use App\Entity\Exceptions\InvalidCodeException;

use App\Entity\Exceptions\CountryNotFoundException;

use App\Entity\Exceptions\DuplicatedCodeException;

class CountryScennarios

{

    public function \_\_construct(

        private readonly CountryRepository $repo

    ) {

    }

    public function SelectAll(): array{

         return $this->repo->GetAll();

    }

    public function GetByCode(string $code): ?Country{

         $country = $this->repo->Get($code);

        if ($country === null) {

            throw new CountryNotFoundException($code);

        }

        return $country;

    }

    public function Add(Country $country): void{

        // выполнить проверку корректности кода

        if (!$this->validate2(code: $country->isoAlpha2) && !$this->validate3(code: $country->isoAlpha3)

            && !$this->validateInt(code: $country->isoNumeric)) {

            throw new InvalidCodeException(

                message: 'validation failed',

            );

        }

        // выполнить проверку уникальности кода

        $code2 = $this->repo->Get(code: $country->isoAlpha2);

        $code3 = $this->repo->Get(code: $country->isoAlpha3);

        $codeInt = $this->repo->Get(code: $country->isoNumeric);

        if ($code2 != null || $code3 != null || $codeInt != null) {

            throw new DuplicatedCodeException();

        }

        // если все ок, то сохранить

        $this->repo->Store(country: $country);

    }

    public function UpdateByCode(string $code, Country $country): void{

        // выполнить проверку наличия

        $updatedCountry = $this->repo->Get(code: $code);

        if ($updatedCountry === null) {

            throw new CountryNotFoundException(notFoundCode: $code);

        }

        // если все ок, то сделать update

        $this->repo->Edit(code: $code, country: $country);

    }

    public function DeleteByCode(string $code): void{

         $country = $this->repo->Get(code: $code);

        if ($country === null) {

            throw new CountryNotFoundException(notFoundCode: $code);

        }

        $this->repo->Delete(code: $code);

    }

    private function validate3(string $code): bool {

        // ^[A-Z]{3}$

        return preg\_match(pattern: '/^[A-Z]{3}$/', subject: $code);

    }

    private function validate2(string $code): bool {

        // ^[A-Z]{2}$

        return preg\_match(pattern: '/^[A-Z]{2}$/', subject: $code);

    }

    private function validateInt(int $code): bool {

        // ^[0-9]{3}$

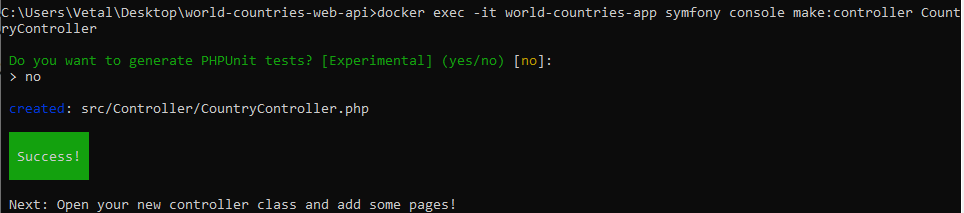
        return preg\_match(pattern: '/^[0-9]{3}$/', subject: $code);

    }

}

6. Добавление контроллера CountryController

docker exec -it world-countries-app symfony console make:controller CountryController



7. Добавление интерфейса для работы с БД (сигнатуры методов)

/app/src/Entity/CountryRepository.php

<?php

namespace App\Entity;

interface CountryRepository

{

    function GetAll(): array;

    function Get(string $code): ?Country;

    function Store(Country $country): void;

    function Edit(string $code, Country $country): void;

    function Delete(string $code): void;

}

8. Имплементация(реализация) интерфейса CountryRepository в app\src\Rdb\CountryStorage с инъекцией объекта SqlHelper

<?php

namespace App\Rdb;

use App\Entity\Country;

use App\Entity\CountryRepository;

use RuntimeException;

use Exception;

class CountryStorage implements CountryRepository

{

    public function \_\_construct(

        private readonly SqlHelper $conn

    ) {

       $conn->pingDb();

    }

    public function GetAll(): array{

        try {

            // создать подключение к БД

            $conn = $this->conn->openDbConnection();

            // подготовить строку запроса

            $queryStr = '

                SELECT short\_name\_f, full\_name\_f, isoAlpha2\_f, isoAlpha3\_f, isoNumeric\_f, population\_f, square\_f

                FROM country\_t;';

            // выполнить запрос

            $rows = $conn->query(query: $queryStr);

            // считать результаты запроса в цикле

            $countries = [];

            while ($row = $rows->fetch\_array()) {

                // каждая строка считается в тип массива

                $country = new Country(

                    shortName: $row[0],

                    fullName: $row[1],

                    isoAlpha2: $row[2],

                    isoAlpha3: $row[3],

                    isoNumeric: intval(value: $row[4]),

                    population: intval(value: $row[5]),

                    square: floatval(value: $row[6])

                );

                array\_push($countries, $country);

            }

            // вернуть результат

            return $countries;

        } finally {

            // в конце в любом случае закрыть соединение с БД если оно было открыто

            if (isset($conn)) {

                $conn->close();

            }

        }

    }

    public function Get(string $code): ?Country{

        try {

            // создать подключение к БД

            $conn = $this->conn->openDbConnection();

            //WHERE

            $param = "isoAlpha3\_f";

            $bind = "s";

            if(intval($code)){

                $param = "isoNumeric\_f";

                $bind = "i";

            }

            else {

                if(mb\_strlen($code)===2) $param = "isoAlpha2\_f";

                elseif (mb\_strlen($code)===3) $param = "isoAlpha3\_f";

            }

            // подготовить строку запроса

            $queryStr = 'SELECT short\_name\_f, full\_name\_f, isoAlpha2\_f, isoAlpha3\_f, isoNumeric\_f, population\_f, square\_f

                FROM country\_t

                WHERE '.$param.' = ?';

            // подготовить запрос

            $query = $conn->prepare(query: $queryStr);

            $query->bind\_param($bind, $code);

            // выполнить запрос

            $query->execute();

            $rows = $query->get\_result();

            // считать результаты запроса в цикле

            while ($row = $rows->fetch\_array()) {

                // если есть результат - вернем его

                return $country = new Country(

                    shortName: $row[0],

                    fullName: $row[1],

                    isoAlpha2: $row[2],

                    isoAlpha3: $row[3],

                    isoNumeric: intval(value: $row[4]),

                    population: intval(value: $row[5]),

                    square: floatval($row[6])

                );

            }

            // иначе вернуть null

            return null;

        } finally {

            // в конце в любом случае закрыть соединение с БД если оно было открыто

            if (isset($conn)) {

                $conn->close();

            }

        }

    }

    public function Store(Country $country): void{

        try {

            // создать подключеник к БД

            $conn = $this->conn->openDbConnection();

            // подготовить запрос INSERT

            $queryStr = 'INSERT INTO country\_t (short\_name\_f, full\_name\_f, isoAlpha2\_f, isoAlpha3\_f, isoNumeric\_f, population\_f, square\_f)

                VALUES (?, ?, ?, ?, ?, ?, ?);';

            // подготовить запрос

            $query = $conn->prepare(query: $queryStr);

            $query->bind\_param(

                'ssssiid',

                $country->shortName,

                $country->fullName,

                $country->isoAlpha2,

                $country->isoAlpha3,

                $country->isoNumeric,

                $country->population,

                $country->square,

            );

            // выполнить запрос

            if (!$query->execute()) {

                throw new Exception(message: 'insert execute failed');

            }

        } finally {

            // в конце в любом случае закрыть соединение с БД если оно было открыто

            if (isset($conn)) {

                $conn->close();

            }

        }

    }

    public function Edit(string $code, Country $country): void{

        try {

            // создать подключеник к БД

            $conn = $this->conn->openDbConnection();

             //WHERE

            $param = "isoAlpha3\_f";

            $bind = "ssii";

            if(intval($code)){

                $param = "isoNumeric\_f";

                $bind = $bind."i";

            }

            else {

                if(mb\_strlen($code)===2) $param = "isoAlpha2\_f";

                elseif (mb\_strlen($code)===3) $param = "isoAlpha3\_f";

                $bind = $bind."s";

            }

            // подготовить запрос INSERT

            $queryStr = 'UPDATE country\_t

                        SET short\_name\_f=?, full\_name\_f=?, population\_f=?, square\_f=?

                        WHERE '.$param.' = ?';

            // подготовить запрос

            $query = $conn->prepare(query: $queryStr);

            $query->bind\_param(

                $bind,

                $country->shortName,

                $country->fullName,

                $country->population,

                $country->square,

                $code,

            );

            // выполнить запрос

            if (!$query->execute()) {

                throw new Exception(message: 'update execute failed');

            }

        } finally {

            // в конце в любом случае закрыть соединение с БД если оно было открыто

            if (isset($conn)) {

                $conn->close();

            }

        }

    }

    public function Delete(string $code): void{

         try {

            // создать подключеник к БД

            $conn = $this->conn->openDbConnection();

             //WHERE

            $param = "isoAlpha3\_f";

            $bind = "s";

            if(intval($code)){

                $param = "isoNumeric\_f";

                $bind = "i";

            }

            else {

                if(mb\_strlen($code)===2) $param = "isoAlpha2\_f";

                elseif (mb\_strlen($code)===3) $param = "isoAlpha3\_f";

            }

            // подготовить запрос INSERT

            $queryStr = 'DELETE FROM country\_t WHERE '.$param.' = ?';

            // подготовить запрос

            $query = $conn->prepare(query: $queryStr);

            $query->bind\_param($bind, $code);

            // выполнить запрос

            if (!$query->execute()) {

                throw new Exception(message: 'delete execute failed');

            }

        } finally {

            // в конце в любом случае закрыть соединение с БД если оно было открыто

            if (isset($conn)) {

                $conn->close();

            }

        }

    }

}

9. Скрипт для первоначальной инициализации БД

mysql\init\init.sql

-- создание таблицы стран

CREATE TABLE country\_t (

id INT NOT NULL AUTO\_INCREMENT,

short\_name\_f NVARCHAR(255) NOT NULL,

full\_name\_f NVARCHAR(255) NOT NULL,

isoAlpha2\_f CHAR(2) NOT NULL,

isoAlpha3\_f CHAR(3) NOT NULL,

isoNumeric\_f INT NOT NULL,

population\_f BIGINT NOT NULL,

square\_f DOUBLE NOT NULL,

--

PRIMARY KEY(id),

UNIQUE(isoAlpha2\_f),

UNIQUE(isoAlpha3\_f),

UNIQUE(isoNumeric\_f),

UNIQUE(full\_name\_f),

UNIQUE(short\_name\_f)

);

-- добавить данные

INSERT INTO country\_t (

short\_name\_f,

full\_name\_f,

isoAlpha2\_f,

isoAlpha3\_f,

isoNumeric\_f,

population\_f,

square\_f

) VALUES

(N'Россия', N'Российская Федерация', 'RU', 'RUS', 643, 146150789, 17125191),

(N'США', N'Соединённые Штаты Америки', 'US', 'USA', 840, 331002651, 9372610),

(N'Китай', N'Народная Республика Китай', 'CN', 'CHN', 156, 1439323776, 9596961),

(N'Индия', N'Республика Индия', 'IN', 'IND', 356, 1380004385, 3287263),

(N'Бразилия', N'Бразильская Республика', 'BR', 'BRA', 76, 212559417, 8515767),

(N'Австралия', N'Содружество Австралия', 'AU', 'AUS', 36, 25499884, 7692024),

(N'Канада', N'Канада', 'CA', 'CAN', 124, 37742154, 9984670),

(N'Германия', N'Федеративная Республика Германия', 'DE', 'DEU', 276, 83783942, 357022),

(N'Франция', N'Французская Республика', 'FR', 'FRA', 250, 65273511, 551695),

(N'Япония', N'Япония', 'JP', 'JPN', 392, 126476461, 377975);

10. Класс SqlHelper с реализацией метода openDbConnection

app\src\Rdb\SqlHelper.php

<?php

namespace App\Rdb;

use mysqli;

use RuntimeException;

use Exception;

class SqlHelper

{

    // openDbConnection - открыть соединение с БД

    public function openDbConnection(): mysqli {

        // зададим параметры подключения к БД

        $host = $\_ENV["DB\_HOST"];

        $port = $\_ENV["DB\_PORT"];

        $user = $\_ENV["USER\_NAME"];

        $password = $\_ENV["USER\_PASSWORD"];

        $database = $\_ENV["DB\_NAME"];

        // создать объект подключения через драйвер

        $connection = new mysqli(

            hostname: $host,

            port: $port,

            username: $user,

            password: $password,

            database: $database,

        );

        // открыть соединение с БД

        if ($connection->connect\_errno) {

            throw new RuntimeException(message: "Failed to connect to MySQL: ".$connection->connect\_error);

        }

        // если все ок - вернуть соединение с БД

        return $connection;

    }

    // pingDb - проверить доступность БД

    public function pingDb() : void {

        // открыть и закрыть соединение с БД

        $connection = $this->openDbConnection();

        $connection->close();

    }

}

11. Exceptions

CountryNotFoundException.php

<?php

namespace App\Entity\Exceptions;

use Throwable;

use Exception;

//исключение не найденной страны

class CountryNotFoundException extends Exception {

    // переопределение конструктора исключения

    public function \_\_construct($notFoundCode, Throwable $previous = null) {

        $exceptionMessage = "country'". $notFoundCode ."' not found";

        // вызов конструктора базового класса исключения

        parent::\_\_construct(

            message: $exceptionMessage,

            code: ErrorCodes::NOT\_FOUND\_ERROR,

            previous: $previous,

        );

    }

}

DuplicatedCodeException.php

<?php

namespace App\Entity\Exceptions;

use Throwable;

use Exception;

// DuplicatedCodeException - исключение дублирующегося кода страны

class DuplicatedCodeException extends Exception {

    // переопределение конструктора исключения

    public function \_\_construct(Throwable $previous = null) {

        $exceptionMessage = "some country code is duplicated";

        // вызов конструктора базового класса исключения

        parent::\_\_construct(

            message: $exceptionMessage,

            code: ErrorCodes::DUPLICATED\_CODE\_ERROR,

            previous: $previous,

        );

    }

}

ErrorCodes.php

<?php

namespace App\Entity\Exceptions;

class ErrorCodes {

    const INVALID\_CODE\_ERROR = 1;

    const NOT\_FOUND\_ERROR = 2;

    const DUPLICATED\_CODE\_ERROR = 3;

}

InvalidCodeException.php

<?php

namespace App\Entity\Exceptions;

use Throwable;

use Exception;

// InvalidCodeException - исключение невалидного кода страны

class InvalidCodeException extends Exception {

    // переопределение конструктора исключения

    public function \_\_construct($message, Throwable $previous = null) {

        $exceptionMessage = "some country code is invalid: ".$message;

        // вызов конструктора базового класса исключения

        parent::\_\_construct(

            message: $exceptionMessage,

            code: ErrorCodes::INVALID\_CODE\_ERROR,

            previous: $previous,

        );

    }

}