**Лабораторная работа “Справочник стран мира”**

**Цель:** научиться разрабатывать WEB-API приложение с использованием PHP, Symfony и MySQL.

**Техническое задание:** реализовать api для выполнения CRUD-операций для справочника стран мира согласно указаниям в ходе работы.

**Стек технологий:** PHP, Symfony, MySQL + mysqli + Workbench, VS Code.

**Ход работы**

1. Создать новое приложение symfony “WorldCountriesDirectoryApp” с помощью команды:

symfony new WorldCountriesDirectoryApp

Запустить созданное приложение и проверить что работает стандартная страничка symfony:

symfony server:start --port=8080

2. Добавить новый контроллер “RootController” с помощью команды:

php bin/console make:controller RootController

При необходимости выполнить установку дополнительных модулей через composer (см. вывод логов).

В данном контроллере добавить 2 обработчика статуса сервера и пинг сервера согласно схеме:

GET /api ->

{

“status”: “server is running”,

“host”: <хост и порт сервера ($request->getHost())>,

“protocol”: <протокол сервера ($request->getScheme)>

}

GET /api/ping -> {“status”: “pong”}

Протестировать обработчики в Postman.

3. Добавить в проекте папку src/Model. src - папка с файлами исходного кода .php. В ней будут создаваться вложенные пакеты и файлы с кодом. Внутри папки Model создать класс Country.php. Внутри данного файла объявить класс-сущность Country, описывающий сущность государства мира:

<php?

namespace App\Model;

class Country {

<описание класса>

}

Поля класса (с примерами значений):

shortName - короткое наименование страны (Россия)

fullName - полное наименование страны (Российская Федерация)

isoAlpha2 - двухбуквенный код страны (RU)

isoAlpha3 - трехбуквенный код страны (RUS)

isoNumeric - числовой код страны - строка (643)

population - население страны - кол-во человек (146 150 789)

square - площадь страны кв. км. (17 125 191)

4. Добавьте в папку Model класс CountryScennarios.php с классом CountryScenarios. В данном классе опишите сигнатуры методов операций для работы со странами, добавив к каждом из методов описание функциональности метода (что делает), что метод принимает на вход и что возвращает, а так же исключения, которые может выбрасывать метод (только для собственных классов исключений).

Подумайте самостоятельно над тем, какие понадобятся исключения для данных операций и добавьте классы этих исключений в папку Model/Exceptions (не забудьте использовать соответствующие пространства имён).

Внутри самих методов пока оставьте тело throw new Exception(‘not implemented’);

Список операций:

**GetAll(): array** - получение списка всех стран.

**Get(code) : Country -** получение списка страны по коду. Можно использовать любой из трех типов кодов, тип кода определяется внутри метода (двухбуквенный, трехбуквенный или числовой).

**Store(country): void** - сохранение новой страны. Можно сохранять страну с уникальными валидными кодами, непустыми уникальными названиями (могут совпадать полное и короткое) и неотрицательным населением и площадью.

**Edit(code, country):** **void -** редактирование страны по коду. можно использовать любой из трех доступных кодов, тип кода определяется в методе (двухбуквенный, трехбуквенный или числовой), можно редактировать все поля кроме полей кодов, также проверять чтобы новые значения полей не противоречили ограничениям из предыдущего метода.

**Delete(code): void -** удаление страны по коду. Можно использовать любой из трех типов кодов, тип кода определяется внутри метода (двухбуквенный, трехбуквенный или числовой).

5. Добавьте в проект новый контроллер CountryController, используя команду аналогичную команде из п.2.

Выполните инъекцию объекта CountryScenarios в данный класс. Все обработчики данного контроллера должны иметь путь /api/country, что можно обеспечить применением атрибута “Route” для всего контроллера.

Методы в контроллер пока не добавлять.

6. В папке src/Model добавьте файл CountryRepository.php с интерфейсом CountryRepository внутри. В данном интерфейсе будут описаны методы хранилища стран. Выполните инъекцию данного интерфейса в класс CountryScenarios.

7. Создайте папку src/Rdb. В ней создайте файл CountryStorage.php с классом CountryStorage, который должен имплементировать интерфейс CountryRepository. Для классов данной папки используйте пространство имен App\Rdb.

8. Время подготовить БД. Для этого предлагается обеспечить работающий экземпляр сервера БД MySQL и клиент для работы с ним (рекомендуется использовать workbench). В корне приложения создайте папку “db” и напишите в ней 2 sql-скрипта:

**country\_ddl.sql** - скрипт создания схемы БД и таблиц. Подумайте над схемой БД - свойствами столбцов и их видом согласно предметной области

**country\_dml.sql** - скрипт заполнения БД тестовыми данными - вставить 5-10 записей (обеспечить очистку БД перед заполнением)

Выполните данные скрипт для подготовки БД и убедитесь в их успешном срабатывании.

9. Добавьте класс SqlHelper в папке src/Rdb. Реализуйте в нем метод openDbConnection, который создает соединение с БД согласно параметрам подключения к БД. Параметры подключения могут быть зашиты в код (как в проекте-примере) либо браться из глобальной переменной $\_ENV (в таком случае данные параметры необходимо указать в файле .env).   
Так же добавьте приватный метод pingDb в данный класс, который выполняет проверку доступности БД путем создания подключения и его закрытия (см. проек-пример). В конструкторе данного класса вызывать метод pingDb.

Выполнить инъекцию объекта SqlHelper в класс CountryStorage, в котором он будет использоваться для создания объекта подключения к БД.

10. Реализовать операции получения всех стран и получения страны по коду в вашем приложении.

Для этого необходимо написать тело методов **GetAll/Get** в классе CountryScenarios, попутно добавляя необходимые методы в интерфейс CountryRepository, которые нужны для методов модели, и имплементируя эти методы в классе CountryStorage с помощью mysqli-классов и методов и SQL-запросов.

Далее в CountryController реализовать обработчики для данных операций и протестировать их. Вызовы api должны соответствовать следующим спецификациям:

**получение всех стран**

GET /api/country -> 200 JSON-массив объектов стран (Country) с полной сериализацией

**получение страны по коду**

GET /api/country/{code} -> 200 объект страны с полной сериализацией

code - url-параметр, в котором передать трехбуквенный, двухбуквенный или числовой код страны, по которому выполняется поиск (метод модели должен валидировать данный код и определять, по какому именно коду выполнять поиск, хранилище в свою очередь может иметь отдельные методы для поиска по каждому из кодов).

Коды ошибок для данного обработчика:

404 - валидный код, но страна по нему не найдена

400 - невалидный код

11. Реализовать операцию добавления новой страны.

Для этого необходимо написать тело метода **Store** в классе CountryScenarios, попутно используя методы хранилища и расширяя его интерфейс при необходимости и имплементируя новые методы в CountryStorage.

Спецификация метода контроллера для операции:

POST /api/country + JSON-объект страны со всеми заполненными полями -> 204 No Content

Модель должна валидировать коды страны (двухбуквенный, трехбуквенный, числовой), проверять чтобы были заполнены неименования (короткое, полное), проверять чтобы население и площадь были неотрицательными а так же проверять уникальность наименований и кодов страны.

Коды ошибок для данного обработчика:

400 - ошибка валидации чего-либо (код, наименование, население, площадь)

409 - ошибка повторяющегося кода или наименования

Протестировать обработчик для всех успешных и неуспешных сценариев.

12. Реализовать операцию редактирования страны.

Для этого необходимо написать тело метода **Edit** в классе CountryScenarios, попутно используя методы хранилища и расширяя его интерфейс при необходимости и имплементируя новые методы в CountryStorage.

Спецификация метода контроллера для операции:

PATCH /api/country/{code} + JSON-объект страны со всеми полями кроме кодов (коды не редактируются, остальное может) -> 200 + JSON-объект страны после редактирования.

code - url-параметр, в котором передать трехбуквенный, двухбуквенный или числовой код страны, по которому выполняется обновление.

Модель должна убедиться в том, что код валидный и страна с таким кодом существует (может быть задан любой из 3х типов кодов). Также модель должна убедиться, что коды страны не меняются.

Коды ошибок для данного обработчика:

400 - невалидный код или невалидные данные для редактирования

404 - не найдена страна по валидному коду

409 - данные для редактирования конфликтуют с существующими данными (дублирование кода и пр.)

Можно на уровне контроллера добавить отдельный класс для передачи обновленных данных, который не будет содержать кодов (чтобы не было возможности коды передать), однако в модель прилетает Country, и в нем коды должны быть заполнены полностью, либо могут быть пустыми и не проверяться - подумайте над этим местом.

Протестировать работу операции.

13. По аналогии со всеми предыдущими пунктами реализовать последнюю операцию удаления.

Спецификация метода контроллера для операции:

DELETE /api/country/{code} -> 204 No Content

code - url-параметр, в котором передать трехбуквенный, двухбуквенный или числовой код страны, по которому выполняется удаление.

Коды ошибок для данного обработчика:

400 - невалидный код

404 - не найдена страна по валидному коду

14. Протестировать еще раз все обработчики. Сделать снимки тестирования API в postman и сохранить данные изображения в директории docs в отдельной папке.

Подготовить диаграмму классов проекта (сохранить в виде картинки в директории docs корня проекта).

**Draw.io - сайт для создания диаграммы классов**

<https://app.diagrams.net/#G1yfyWnwLxhrM50j-OWytZ0R5pV-69mrzA#%7B%22pageId%22%3A%22C5RBs43oDa-KdzZeNtuy%22%7D>

Опубликовать проект на github (используйте интерфейс VS Code и терминал, репозиторий уже проинициализирован).

Можно отвести ветку dev с изменениями и открыть pull request в master для получения ревью от преподавателя (по желанию).

В ответ на ДЗ по данной работе отправить ссылку на github-репозиторий с полученным проектом, скриптами БД, диаграммой классов и снимками работы.