Лабораторная работа №7 «Инфраструктура: хранение и доставка. Межсервисное взаимодействие»

Цель работы

Познакомиться с практической реализацией принципа инверсии зависимостей, а также протоколами межсервисного взаимодействия

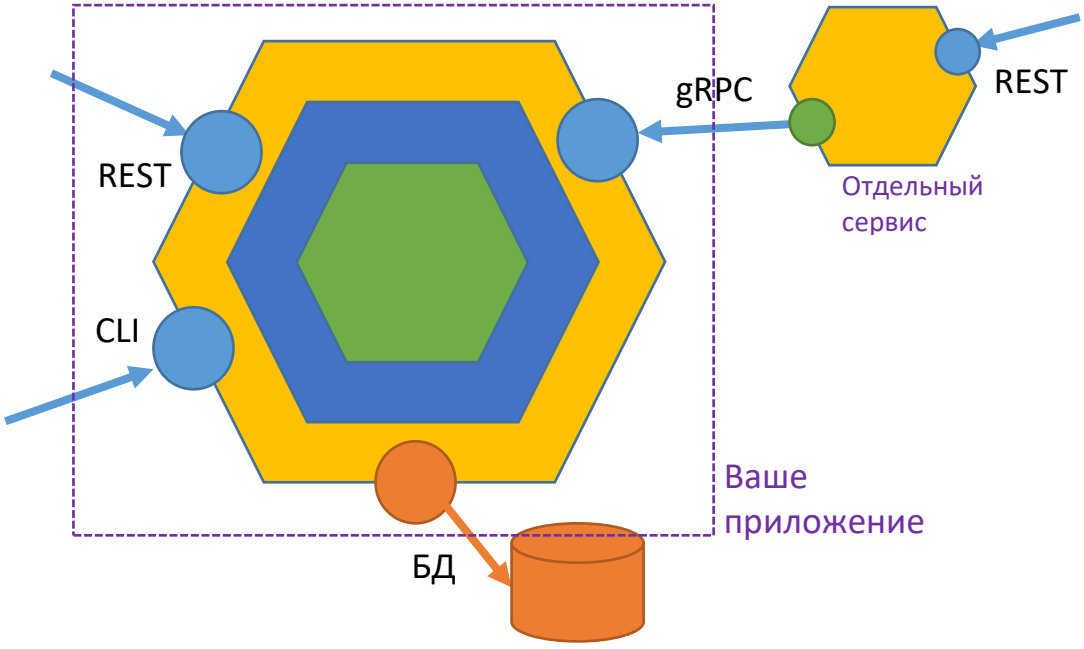
Задание для выполнения

Реализуйте механизмы хранения (persistence) для ранее разработанного приложения – технологию, фреймворк, ORM выберите сами. Это могут быть реляционная БД, NoSQL, InMemory, файловая система, Redis и т.д.

Реализуйте минимум два механизма доставки – способов вызова функций вашего приложения. Например, REST и командная строка.

Отдельно реализуйте микросервис, который не будет выполнять никакой полезной работы, кроме вызова какой-либо функции вашего приложения по протоколу gRPC (третий способ доставки для вашего приложения), используйте Google Protobuf. Большим плюсом станет, если данный микросервис будет реализован не на PHP.

Итоговая архитектура должна получиться примерно такой:



Теоретические сведения

Вспомните всё, что изучили ранее.

Persistence

О хранении было уже сказано достаточно много. Репозиторий – паттерн хранения, относится к доменной модели в виде интерфейса, а реализация – на уровне инфраструктуры, что позволяет инвертировать зависимость и удовлетворить dependency rule.

Таким образом, с уровня приложения сервисы вызывают методы интерфейсов репозиториев доменного уровня для получения агрегатов или их сохранения. Реальная же реализация этих интерфейсов адаптирована под конкретную технологию хранения, вид БД, систему ORM и т.д.

Наиболее употребительными являются типы репозиториев: **InMemory –** без реального сохранения, все данные в массивах оперативной памяти, т.е. в рамках одного вызова, используется обычно для нужд тестирования; **MySQL, Postgres** и т.п. – в соответствующей СУБД, с минимальной прослойкой между приложением и СУБД, типа PDO; **Doctrine, Hibernate** – соответствуют используемой ORM.

Файловая структура инфраструктурного уровня обычно непосредственно сообщает, какие репозитории для каких доменных моделей (агрегатов) реализованы.

Напоминаем, что репозитории нужны только для агрегатов, и оперируют целыми агрегатами. Т.е. не должно быть в репозитории заказов отдельных методов для загрузки строк заказа – весь агрегат грузится целиком методом `get` или аналогичным.

Пример файловой структуры инфраструктурного уровня (он повторяет обычно доменный уровень на каком отрезке) для агрегата Invoice модуля Payment:

```
Payment
  Domain
    Model
      Invoice
        Invoice.php
        InvoiceRepositoryInterface.php
  Infrastructure
    Domain
      Model
        Invoice
          InMemory
            InvoiceRepository.php
          Doctrine
            InvoiceRepository.php
```

Delivery

Доставка как реализация вызова вашего приложения (сервиса) внешним миром фигурирует на инфраструктурном уровне отдельно от хранения.

```
Payment
  Infrastructure
    Delivery
      Rest
        PaymentService.php
      Grpc
        PaymentService.php
      Cli
        PaymentService.php
```

Все три представленных `PaymentService` являются точками входа в приложение (сервис) и могут (но не обязаны) реализовывать одни и те же вызовы. Например, и *Rest*-, и *Cli*-реализации могут иметь одинаковые методы, только *Rest* – для фронтенда и одиночных вызовов, а *Cli* – для пакетного вызова администратором.

Отдельно стоит отметить gRPC сервис. Он является реализацией интерфейса, который получен автоматически компиляцией определений Protobuf (про то, как оформить и скомпилировать в различные языки – [1]). Технически это может быть сделано следующим образом:

```
Core
  proto
    payment
      payment_service.proto
  Service
    Payment
      PaymentService.php   // интерфейс
Payment
  Infrastructure
    Delivery
      Grpc
        PaymentService.php // реализация
```

В коде же это будет выглядеть примерно так:

```php
<?php
namespace My\Payment\Infrastructure\Delivery\Grpc;
```

```
class PaymentService implements \Core\Service\Payment\PaymentService
{
    //...
}
```

Список использованных источников

1. Protocol Buffers | Google Developers [Электронный ресурс]. – Режим доступа: https://developers.google.com/protocol-buffers. – Дата доступа: 02.03.2020 г.