CLARYFI

Where innovation meets clarity

Условие:

- Есть некая условно компания “ZOO”, которая в данный момент работает с двумя перевозчиками:
  - TransCompany
  - PackGroup
- У каждого перевозчика всегда своя формула расчета стоимости доставки посылки (для простоты все цены будут всегда в EUR):
  - **TransCompany** до **10 кг** берет **20 EUR**, все что свыше **10 кг** - **100 EUR**
  - **PackGroup** за каждый **1 кг** берет **1 EUR**
  - в будущем будут добавляться другие новые перевозчики со своей формулой расчета (это нужно учесть).

Задача:

Необходимо описать ООП архитектуру на PHP из методов, классов, модулей, апи-контроллеров и т.д. для работы с перевозчиками на предмет получения стоимости доставки по каждому из указанных перевозчиков, согласно их формулам. При разработке нужно учесть, что количество перевозчиков со временем может возрасти. И у новых добавленных перевозчиков формулы расчета стоимости будут другими. **Подсказка для реализации:** наследование, абстрактные классы, класс-сервис для расчета стоимости, который будет принимать класс транспортной службы и входные данные от клиента (в данном случаи массу посылки) и тд.

Ожидаемый результат:

- На фронте web страничка с простыми полями ввода: **массы посылки, селект поле** с выбором slug (или ID, не важно) перевозчика и кнопкой **“calculate price”.**
- На бекенде - API контроллер, который будет принимать массу посылки и slug (или ID, не важно) перевозчика и в ответ отдавать рассчитанную стоимость в EUR.

CLARYFI

Where innovation meets clarity

Требования:

Задание нужно сделать на **php фреймворке Symfony**, так как в будущему сотруднику у нас предстоить работать с ним на бекенде и немного с фронтенд фреймворком Vue.js (бекенд сопровождает простенькую админку на Vue.js. Нам важно понимать Ваше знание или отсутствие знания Vue.js), то желательно использовать для отрисовки полей ввода Vue.js (без фанатизма, все простенько), а для бекенд части – Symfony. Также нужно будет работать с системой контейнеризации **Docker**. Поэтому в идеале всю реализацию завернуть в Docker, то есть создать ngnix-контейнер, php-контейнер, ui-контейнер и все это соркестрировать через **docker-compose.yml. PHPunit тесты** приветствуются, так как в будущей работе их также сотруднику придется писать.