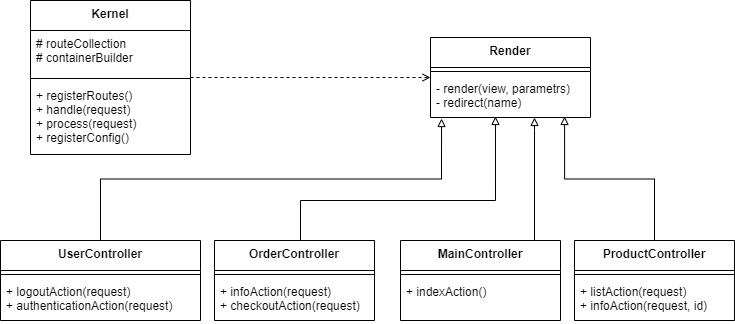
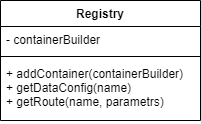
**1. Найти и указать в проекте Front Controller и расписать классы, которые с ним взаимодействуют.**



**2. Найти в проекте паттерн Registry и объяснить, почему он был применён.**



Данный паттерн применен для централизованного доступа к данным конфигурации и коллекции роутов.

**3. Добавить во все классы Repository использование паттерна Identity Map вместо постоянного генерирования сущностей.**

Реализацию самого паттерна взял из методички

**class** IdentityMap  
{  
 **private $identityMap** = [];  
  
 **public function** add($obj)  
 {  
 $key = $this->getGlobalKey(get\_class($obj), $obj->getId());  
  
 $this->**identityMap**[$key] = $obj;  
 }  
  
 **public function** get(**string** $classname, **int** $id)  
 {  
 $key = $this->getGlobalKey($classname, $id);  
  
 **if** (**isset**($this->**identityMap**[$key])) {  
 **return** $this->**identityMap**[$key];  
 }  
 **return null**;  
 }  
  
 **private function** getGlobalKey(**string** $classname, **int** $id)  
 {  
 **return** sprintf(**'%s.%d'**, $classname, $id);  
 }  
}

Подключение паттерна Identity Map на примере одного метода

**public function** search(**array** $ids = []): **array**{  
 **if** (!count($ids)) {  
 **return** [];  
 }  
  
 $identityMap = **new** \IdentityMap();  
  
 $productList = [];  
  
 **foreach** ($this->getDataFromSource([**'id'** => $ids]) **as** $item) {

$model = $identityMap->get(get\_class($this), $item[**'id'**]);

**if** ($model == **null**) {  
 $model = **new** Entity\Product($item[**'id'**], $item[**'name'**], $item[**'price'**]);  
 $productList[] = $model;  
 $identityMap->add($model);  
 } **else** {  
 $productList[] = $model;  
 }  
 }  
  
 **return** $productList;  
}