Algorithm 2: Algorithm to allocate reactor functions to a cluster

```
ALLOCATE_REACTOR_FUNCTION(RAF)
foreach $f \in RAF$ do
    $f_{tb} \leftarrow$ tables accessed and manipulated by $f$;
    if $f_{tb}$ contains only one table then
        $i \leftarrow$ cluster that holds table $\in f_{tb}$;
        allocate $f$ to cluster $i$;
    else
        $max\_access\_freq \leftarrow 0$;
        $alloc\_cluster \leftarrow \varnothing$;
        foreach $tb \in f_{tb}$ do
            $access_{tb} \leftarrow$ access frequency of the respective interface of $tb$;
            if $access_{tb} > max\_access\_freq$ then
                $max\_access\_freq \leftarrow access_{tb}$;
                $alloc\_cluster \leftarrow$ cluster that holds $tb$;
        allocate $f$ to cluster $alloc\_cluster$;
```

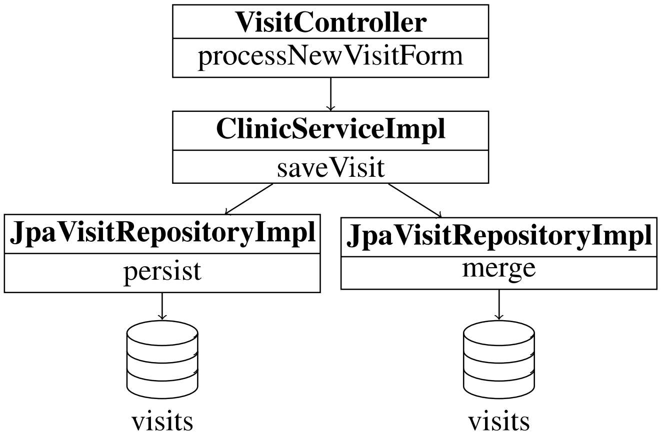

Figure 4. Dependency graph of *_/visits/new_* interface POST operation in Petclinic

Petclinic, e.g., the insertion rate into the *pets* table cannot be lower than of owners (a *pet* cannot exist without an *owner*) or table *visits* must incur the highest access frequency. It is worthy to mention that access frequency in our model ranges from 1 to 100.

Table coupling identification. The entity-relationship (ER) diagram for the Petclinic application is depicted in Figure 5. As can be seen, the relationships with coupling equal to 1 are: types and pets, owners and pets, and visits and pets.

Reactor table identification. In order to execute the model, the parameter Q was set to 200, corresponding to the sum for the resource with the highest access frequencies (*visits*). We aim to distribute workload among reactors, avoiding two or more data-intensive entry points to be allocated in the same reactor type. Figure 6 exhibits the result of the optimization allocating tables to clusters. The result of the allocation of interfaces to clusters can be accessed online [4].