# Language Detection Web Application – Duck Soft Works § Co.

# US 1 – As an Administrator I want to be able to consult, create and delete an item from the blacklist.

## 1. Requirements Engineering

### 1.1. Customer Specifications and Clarifications

### Q: The Blacklist should be created by the Administrator, should it exist already as a file to be consulted by the application or should we take advantage of an existent Blacklist API and only consult it? R: The blacklist is managed by an Information Manager (Admin). It is a concept within the project itself and therefore created as so. You should make use of Bootstrap to create the first insertion.

### 1.2. Acceptance Criteria

The URLs in the Blacklist must be in a valid URL format.

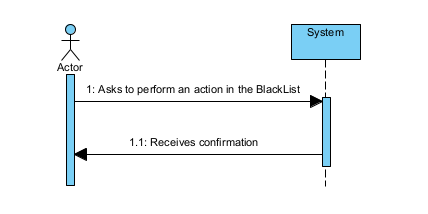
### 1.3. Found out Dependencies

To be able to create a task, a user must provide a valid text url that is not contained in the BlackList. For that reason, the url must always be compared to all the elements that list to make sure it has permission to continue to the language detection task.

### 1.4. Input and Output Data

The input to insert a certain url into the Blacklist is given by the Administrator. At this point, we are still not using roles. Therefore, the only url counting in the Blacklist is the one given by the PO that was to be imputed by bootstrap.

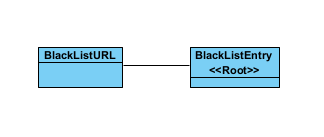
### 1.5. System Sequence Diagram (SSD)



These actions can be to create or to delete a new BlackList item, or consult all the BlackList items already previously inserted..

## 2. OO Analysis

### 2.1. Relevant Domain Model Excerpt

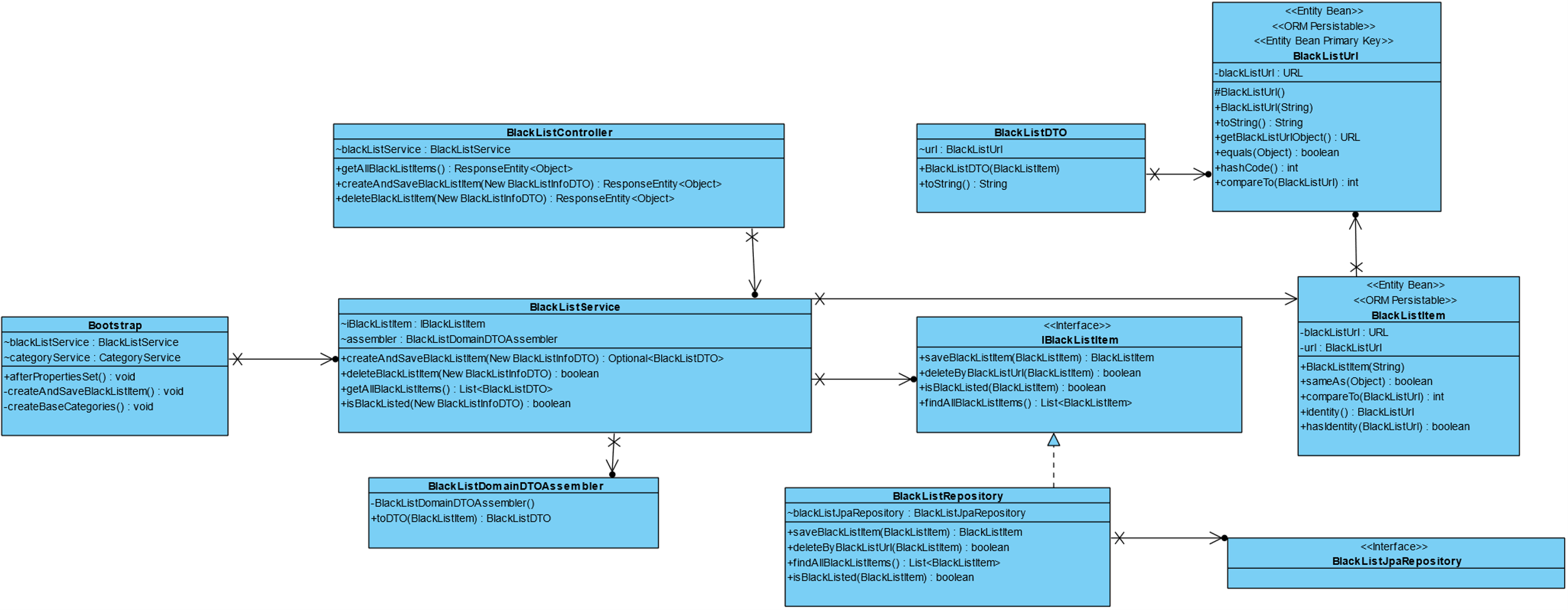


## 3. Design - User Story Realization

## 3.1. Sequence Diagram (SD)

For these sequence diagrams, and because it involves actions that are very similar to other User Stories for the application, please check Generic Create, Generic Delete and Generic FindAll diagrams that are present in the Team’s Report.

## 3.2. Class Diagram (CD)



# 4. Tests

This is a mere example of one of the tests performed for this use case.

This in particular is for a successful deletion of a BlackList item.

@Test  
void shouldSuccessefullyDeleteABlackListItem() throws MalformedURLException {  
 //Arrange  
 String Url = "https://stackoverflow.com/first";  
 BlackListItem blackListItem1 = new BlackListItem(Url);  
 NewBlackListInfoDTO newBlackListInfoDTOitem1 = new NewBlackListInfoDTO(Url);  
 //Act  
 Mockito.*when*(iBlackListItem.deleteByBlackListUrl(blackListItem1)).thenReturn(true);  
 //Assert  
 Assert.*assertTrue*(blackListService.deleteBlackListItem(newBlackListInfoDTOitem1));  
}

# 5. Construction (Implementation)

There are still some exceptions that are not treated as we would like to be when thrown.

In this Sprint it was not possible to manage the time to be able to do it so, but it is something to deal with in the next sprint for sure.

# 7. Observations

According to information retrieved from the client and considering all the tasks that we were attempting to get done in this sprint, we are aware that there are things to improve but we think that this User Story is well under way.