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?

A: 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.

Q: Does a Blacklist block the URL itself or should it block the domain/ IP address as well?

A: The Blacklist is an address list. All addresses/subpaths that derive from it should also be blacklisted.

1.2. Acceptance Criteria

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

A URL cannot be blacklisted if it has been so already or if a URL containing the path for the URL being inserted has already been blacklisted.

1.3. Found out Dependencies

N/A - No found out dependencies for this US.

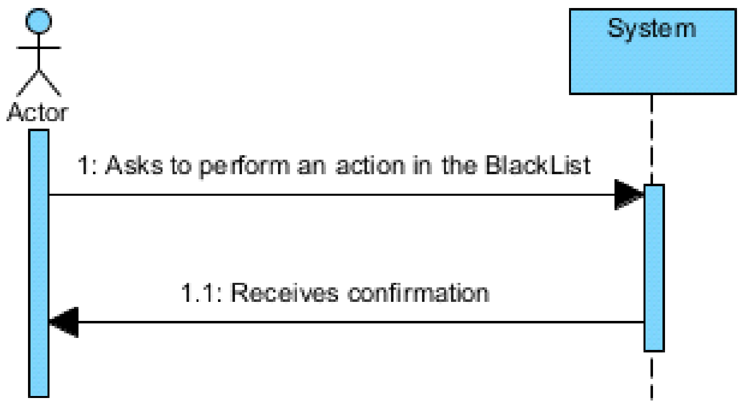
1.4. Input and Output Data

The input to insert a certain URL into the Blacklist is given by the administrator. Therefore, insertion of any data into the system related to blacklist functionalities is only possible if a user with an admin role has been previously authenticated and has authorities to do so.

Currently, there is only one administrator in the system, created by a data bootstrap. No manual insertion is possible as the system is not yet prepared to do so.

Outputted data includes responses that denote success, or otherwise, upon insertion or deletion a blacklist item, as well as a list of all blacklisted items previously created.

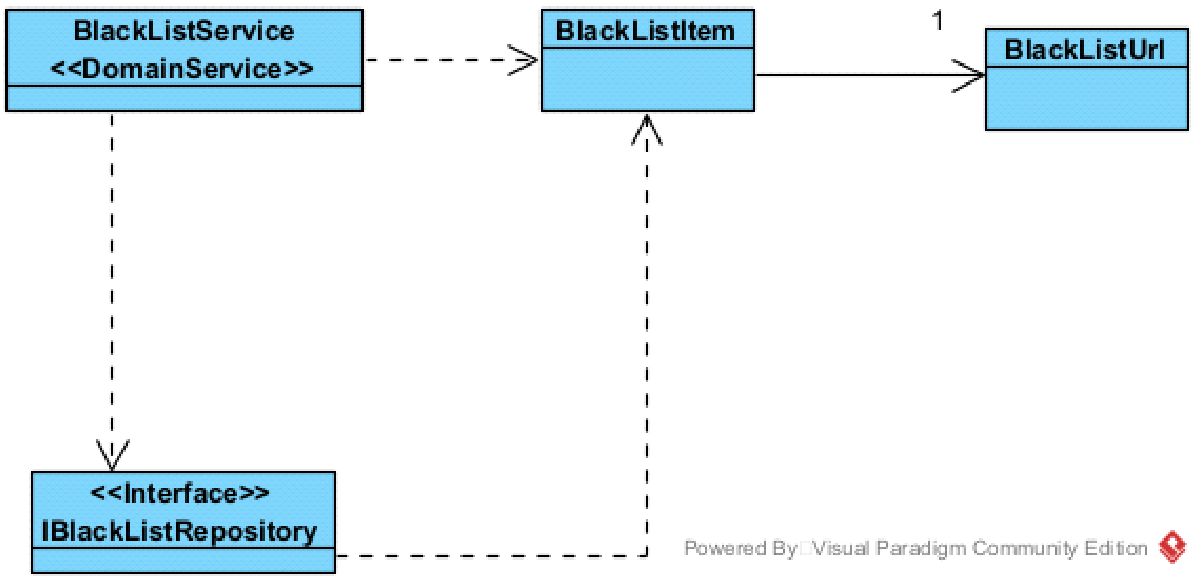
1.5. System Sequence Diagram (SSD)


System
Actor
1: Asks to perform an action in the BlackList
1.1: Receives confirmation

This SSD illustrates the actions performed by the admin actor. Those actions can be to create a new blacklist item, consult all items previously created and saved, or delete an existing blacklist item.

2. OO Analysis

2.1. Relevant Domain Model Excerpt



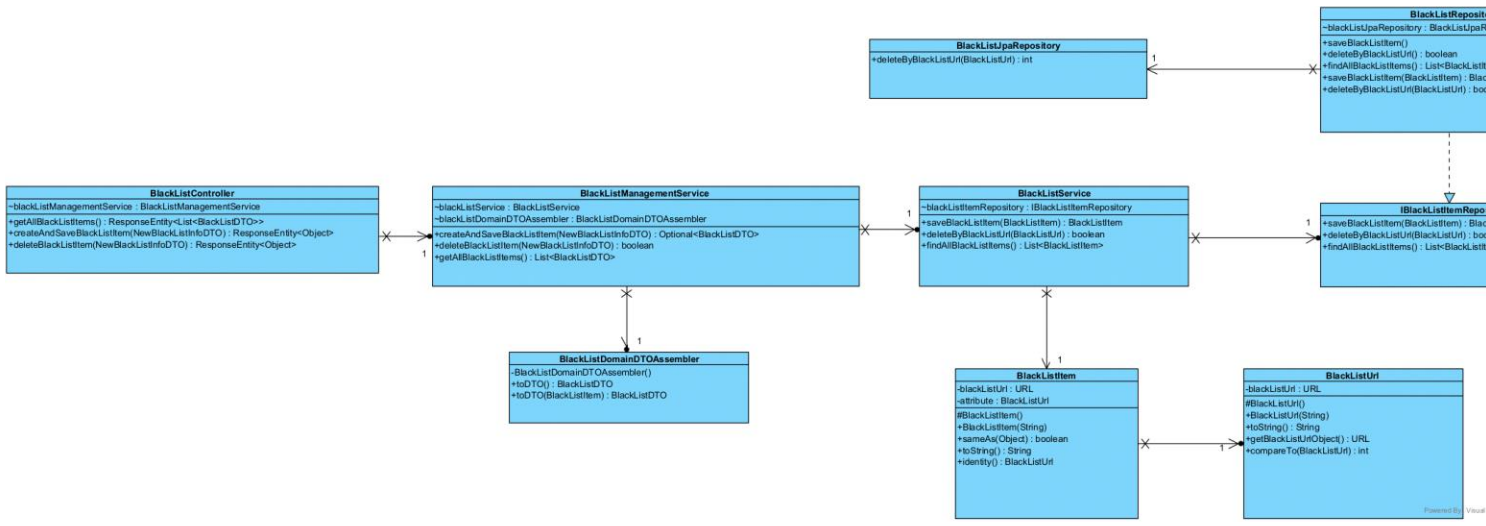
An excerpt of the general domain model, adapted for this US. The represented concepts are the BlackListItem aggregate root and it's BlackListURL value object, the BlackListService domain service and the IBlackListRepository

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)

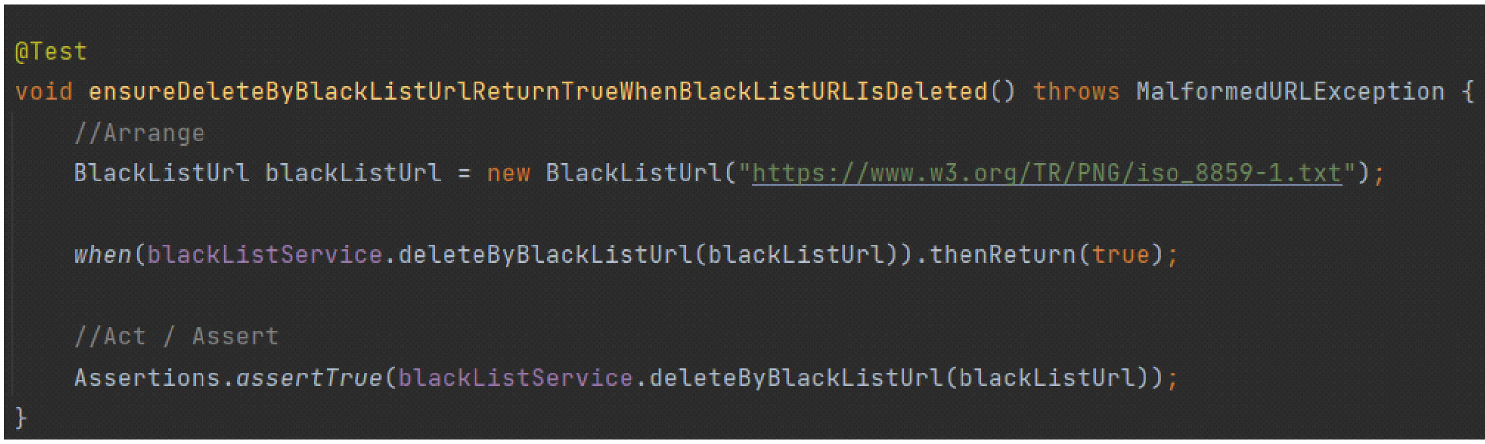

BlackListReposit
~blackListJpaRepository : BlackListJpaR
+saveBlackListItem()
+deleteByBlackListUrl() : boolean
+findAllBlackListItems() : List<BlackListIt
+saveBlackListItem(BlackListItem) : Blac
+deleteByBlackListUrl(BlackListUrl) : boo
BlackListJpaRepository
+deleteByBlackListUrl(BlackListUrl) : int
1
BlackListController
~blackListManagementService : BlackListManagementService
+getAllBlackListItems() : ResponseEntity<List<BlackListDTO>>
+createAndSaveBlackListItem(NewBlackListInfoDTO) : ResponseEntity<Object>
+deleteBlackListItem(NewBlackListInfoDTO) : ResponseEntity<Object>
1
BlackListManagementService
~blackListService : BlackListService
~blackListDomainDTOAssembler : BlackListDomainDTOAssembler
+createAndSaveBlackListItem(NewBlackListInfoDTO) : Optional<BlackListDTO>
+deleteBlackListItem(NewBlackListInfoDTO) : boolean
+getAllBlackListItems() : List<BlackListDTO>
1
BlackListService
~blackListItemRepository : IBlackListItemRepository
+saveBlackListItem(BlackListItem) : BlackListItem
+deleteByBlackListUrl(BlackListUrl) : boolean
+findAllBlackListItems() : List<BlackListItem>
1
IBlackListItemRepo
+saveBlackListItem(BlackListItem) : Blac
+deleteByBlackListUrl(BlackListUrl) : boo
+findAllBlackListItems() : List<BlackListIt
1
BlackListDomainDTOAssembler
-BlackListDomainDTOAssembler()
+toDTO() : BlackListDTO
+toDTO(BlackListItem) : BlackListDTO
1
BlackListItem
-blackListUrl : URL
-attribute : BlackListUrl
#BlackListItem()
+BlackListItem(String)
+sameAs(Object) : boolean
+toString() : String
+identity() : BlackListUrl
1
BlackListUrl
-blackListUrl : URL
#BlackListUrl()
+BlackListUrl(String)
+toString() : String
+getBlackListUrlObject() : URL
+compareTo(BlackListUrl) : int
Powered By Visual

A class diagram excerpt representing all of the classes and relevant attributes and methods that play a part on admin blacklist management related functionalities.

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.



```java
@Test
void ensureDeleteByBlackListUrlReturnTrueWhenBlackListURLIsDeleted() throws MalformedURLException {
    //Arrange
    BlackListUrl blackListUrl = new BlackListUrl("https://www.w3.org/TR/PNG/iso_8859-1.txt");

    when(blackListService.deleteByBlackListUrl(blackListUrl)).thenReturn(true);

    //Act / Assert
    Assertions.assertTrue(blackListService.deleteByBlackListUrl(blackListUrl));
}
```

5. Construction (Implementation)

We recurred to a repository interface implementation to allow different persistence possibilities.

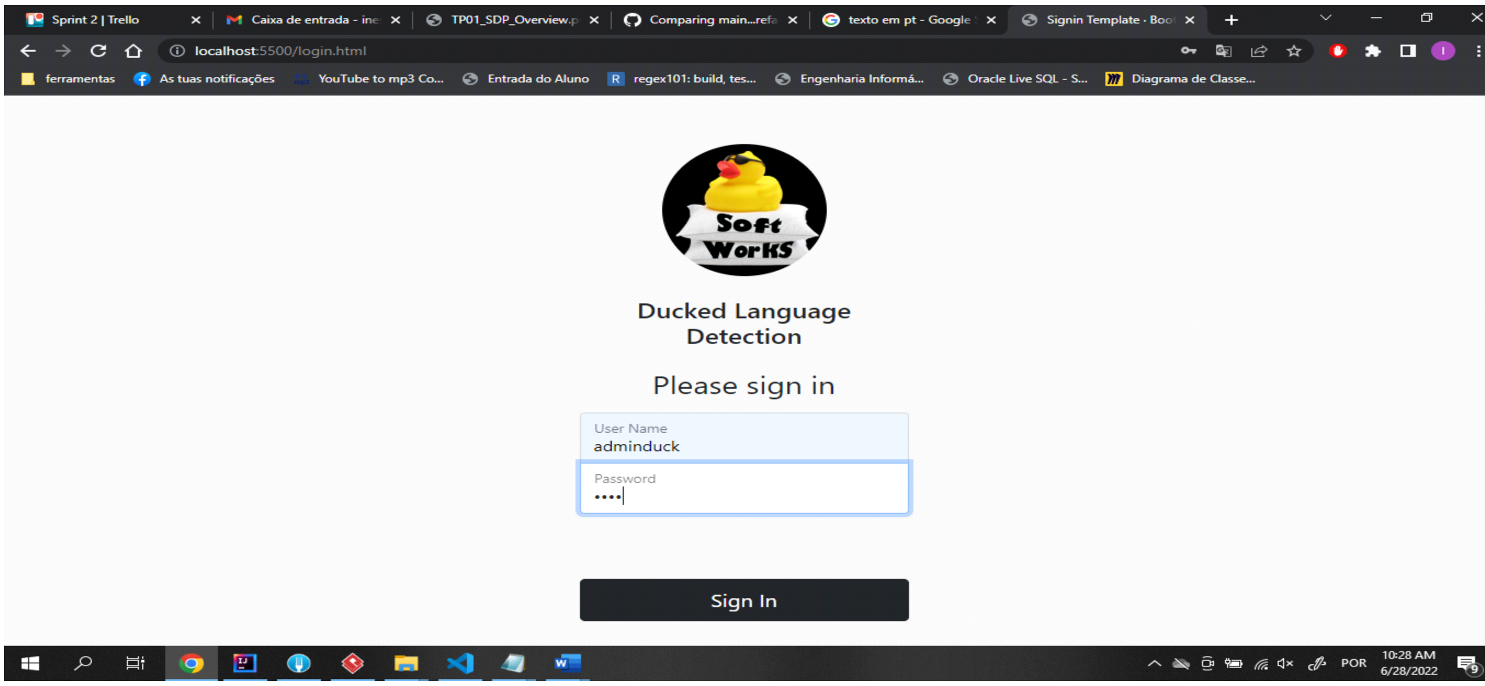
Only methods that reflected the administrator needs were made available. These rules out possibilities like searching for a specific BlackListUrl as no method exists for it.

The System is built so no BlackListItem can be created if the provided BlackListUrl is not compliant with HTTP protocol, has empty spaces only, numbers only or is null. This is so that it can be used in US that have a dependency with this US.

6.Integration and Demo

For blacklist related functionalities, it is then necessary to be logged in with a user with administrator permissions, so that creation, search and deletion can be accomplished.

We made an integration with one frontend application that can be used in conjunction with the backend application.

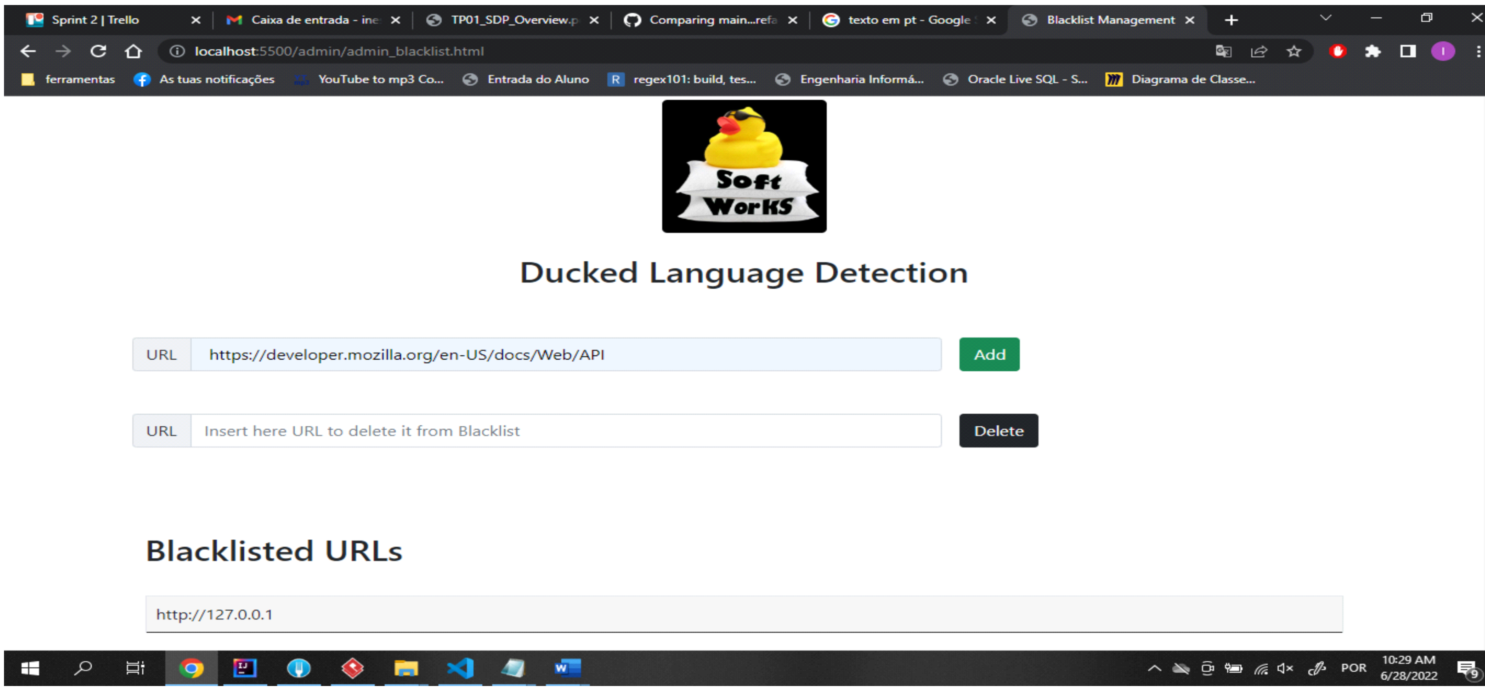


Ducked Language Detection

Please sign in

User Name: adminduck

Password: ••••



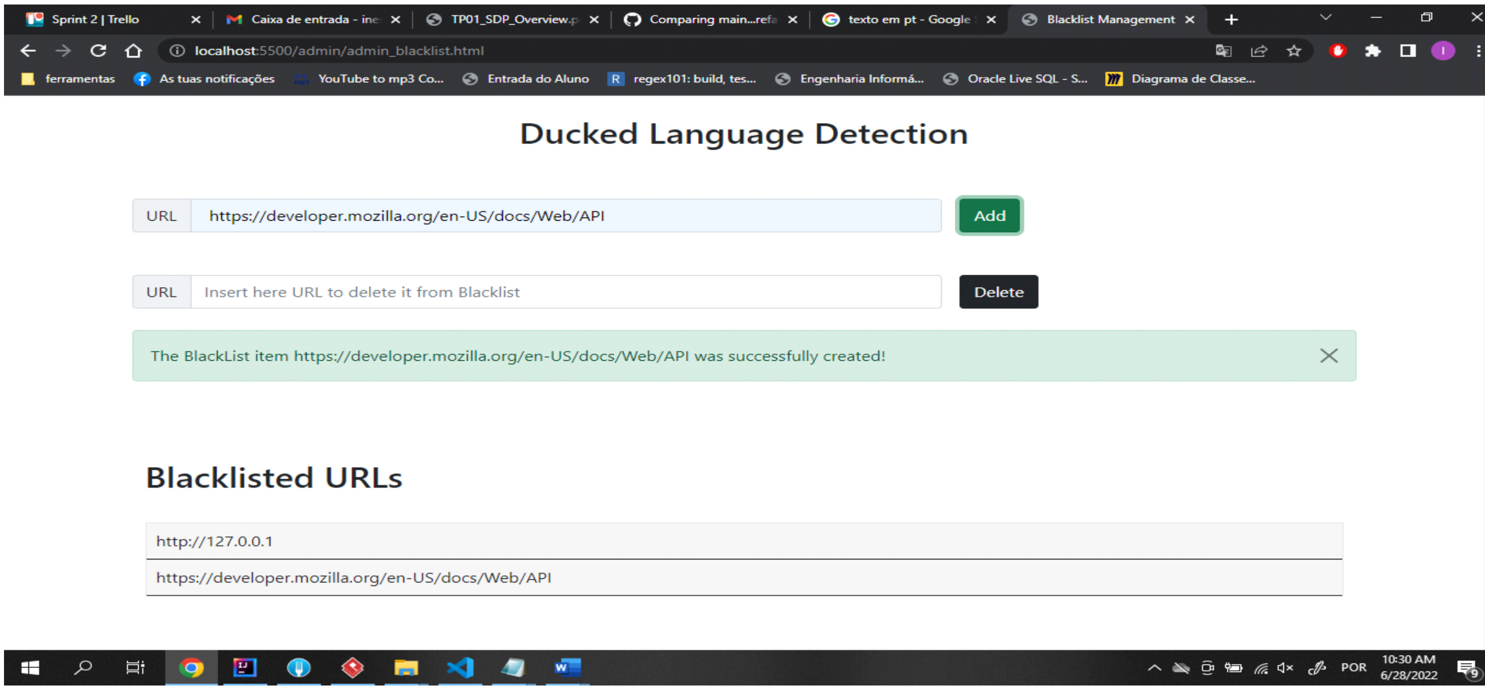
Ducked Language Detection

URL: https://developer.mozilla.org/en-US/docs/Web/API

URL: Insert here URL to delete it from Blacklist

Blacklisted URLs

http://127.0.0.1



Ducked Language Detection

URL https://developer.mozilla.org/en-US/docs/Web/API Add

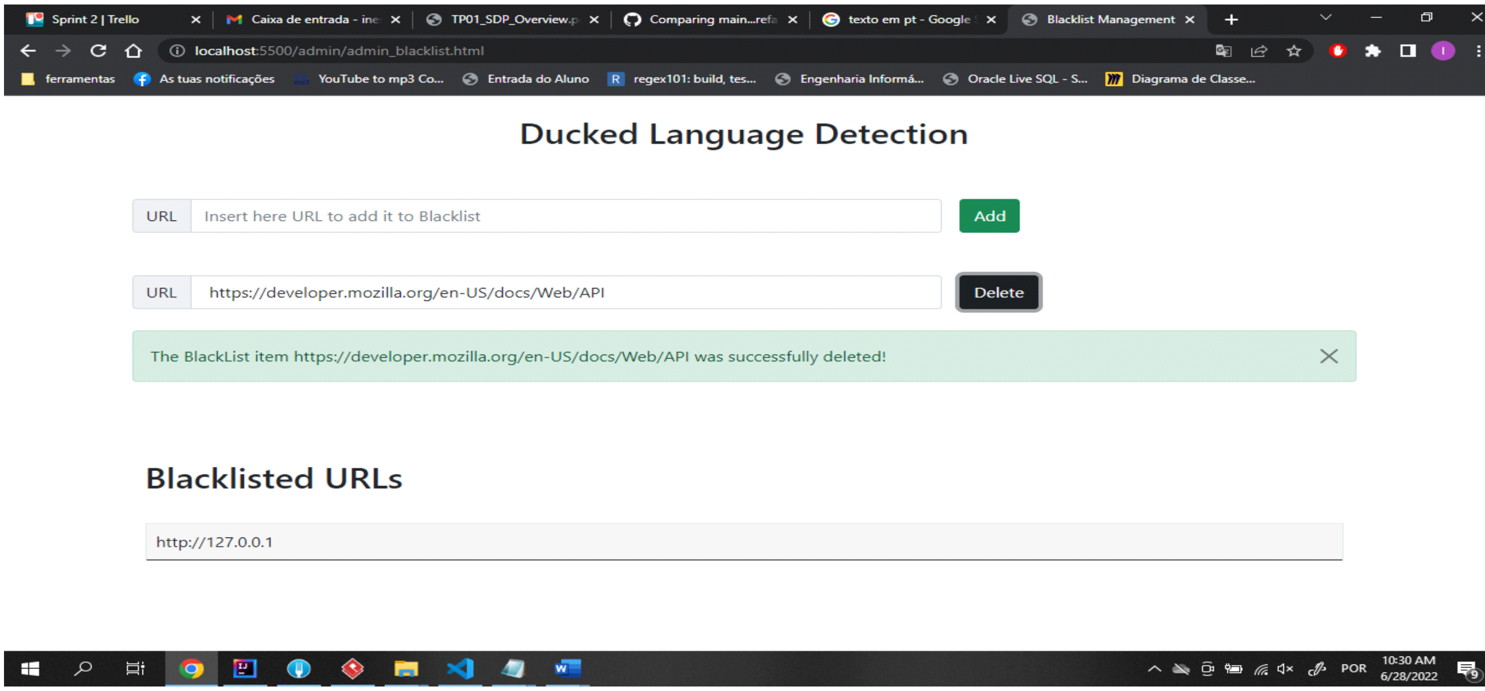
URL Insert here URL to delete it from Blacklist Delete

The BlackList item https://developer.mozilla.org/en-US/docs/Web/API was successfully created!

Blacklisted URLs

http://127.0.0.1

https://developer.mozilla.org/en-US/docs/Web/API



Ducked Language Detection

URL Insert here URL to add it to Blacklist Add

URL https://developer.mozilla.org/en-US/docs/Web/API Delete

The BlackList item https://developer.mozilla.org/en-US/docs/Web/API was successfully deleted!

Blacklisted URLs

http://127.0.0.1

7. Observations

All things taken into account, realization of this US was accomplished successfully.

Nonetheless, there are still some considerations:

-The usage of a BlackListUrl concept is due to constraints in terms of implementation and not necessarily due to a concept that was identified by itself during analysis (refer to the general report for further information).

-At some point, a registry for further users (administrators included) was a concern. The dev team has opted to rule this out since there were no specific requirements for this and the client has specified admin creation through bootstrapped data.

-A feature that blocked the domain of a provided BlackListUrl was something that the team pondered at first, but later found out that should be discarded due to client clarifications.

-Appropriate responses to the HTTP requests, on the event that the action performed on the system was unsuccessful, have yet to be properly implemented. As an alternative for the moment, however, we attempt to communicate the possible failure causes with a generic approach.