**EAD2 CA2 Report – We.Trade**

X00131398 - Toba Samuel

X00127285 - Aaron O’Connor





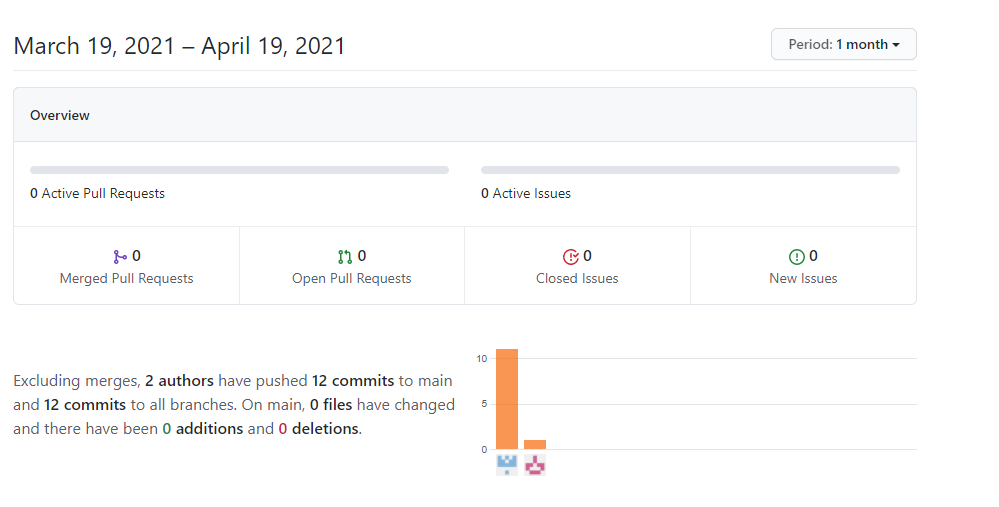
**Introduction**

We decided for our Enterprise Applications Development 2 2nd CA we would create an app where users can see sellers and the clothes of the sellers on the app. The idea is that users would be able to see a list of clothes and also a list of sellers. Users should be able to put clothes up for sale with information added for the item such as the name, price, name of the brand the size and the colour. The clothes are connected to the user by id. You can see how good a seller is by their user rating which is a variable in the seller object. You can also see how many other clothes they’ve sold in the past. Users can use the number of clothes sold and the rating of the seller to judge whether the seller is a good seller or not. Also, users can see the location of the sellers as this is also a variable. This will make it easier for users to decide if the clothes are suitable for them at the location that it is located at.

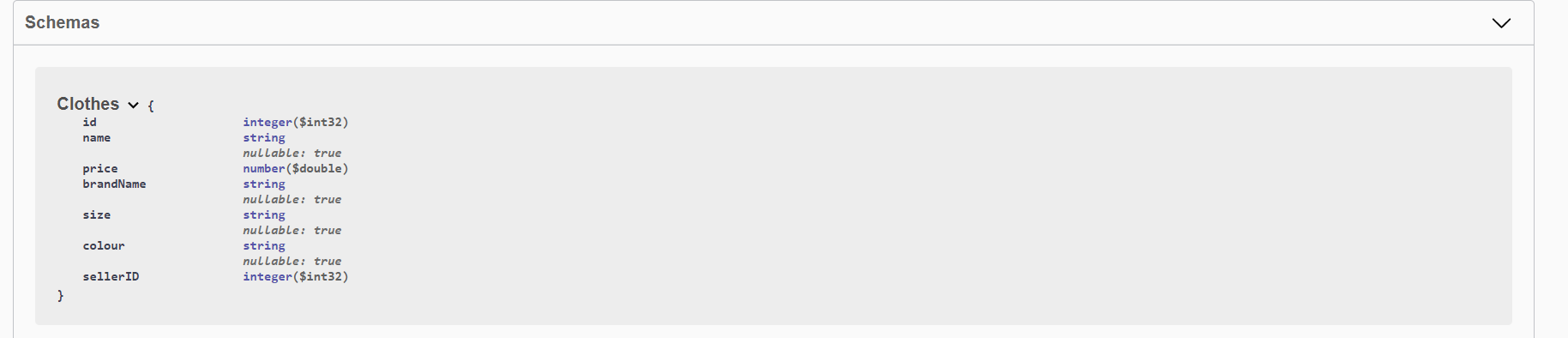
**Github**

The link to the github repository is: <https://github.com/tsam137/EAD2_CA2>

Github was the website we used to store our repository. This helped us to work on various parts of the project without overlapping. We used the repository as a way to keep track of the work we had done. This was a good way to see what was done and what was yet to be done. We committed our work whenever we finished a task we had set out to do. To avoid errors being committed to the repository and potentially affecting the files, we only really committed when we were finished a task and it was working. This means there are not a huge amount of commits but we did this to ensure there would be little error in our commits.

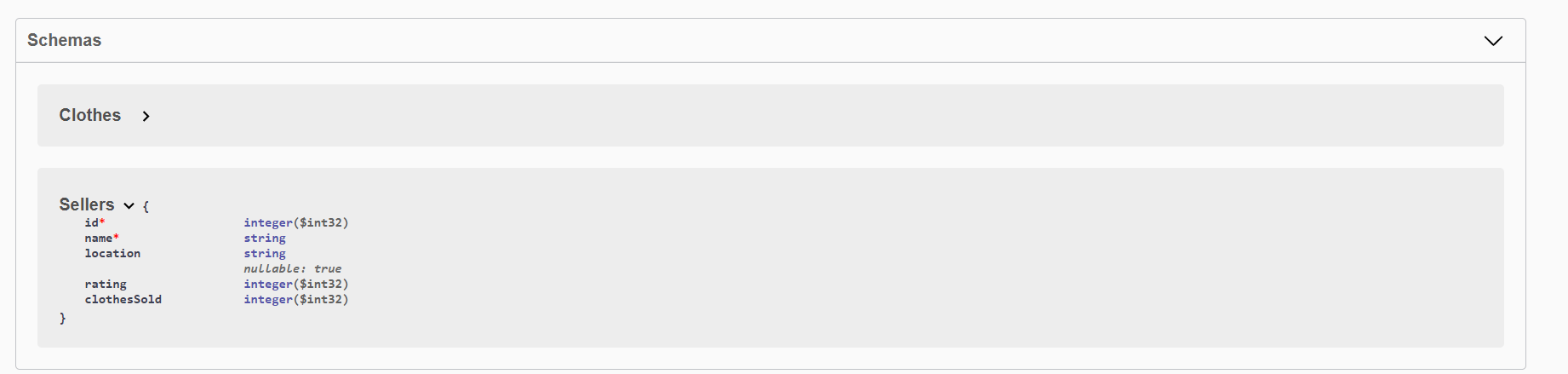


**Database – Schema**

**Clothes Schema**

In this schema we can see that we have 7 variables. id, name, price, brandName, size, colour and sellerID. ID is an int that keeps track of the identification number of the item of clothing. Name is a string and it holds the name that the seller has given to the clothing, usually “Jacket” or “Tracksuit”. Price is a double and it represents the price of the item of clothing. BrandName is a string and it holds the name of the brand of the item of clothing. Size is a string and it holds the size of the item of clothing. Colour is also a string and it holds the colour of the item of clothing. SellerID is an int and it holds the sellers ID, this links with the sellers in the Seller schema. It is a foreign key and is what can be used to reference the sellers for the specific item of clothing.

**Seller Schema**

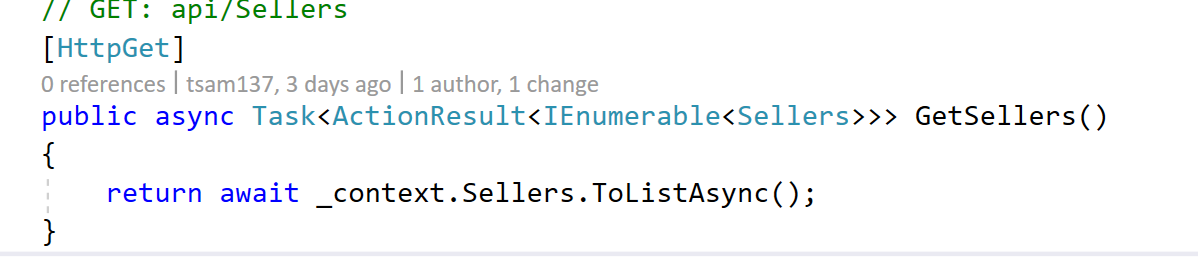
****

In this schema we can see that we have 5 variables. id, name, location, rating, clothesSold. ID is an int that keeps track of the identification number of the seller of the clothing. Name is a string and it holds the name of the seller. Location is a string that stores the location of the seller in question, it is usually “Naas”, “Tallaght”. Rating is an int that stores the rating of the customer, for example “4”. This rating is out of 5 and is a great indicator as to the credibility of a seller. clothesSold is also an int that stores the number of clothes sold by a specific seller. This is also a good indicator as to the credibility of the seller as it clearly shows how new they are to the market or if they are an experienced seller.

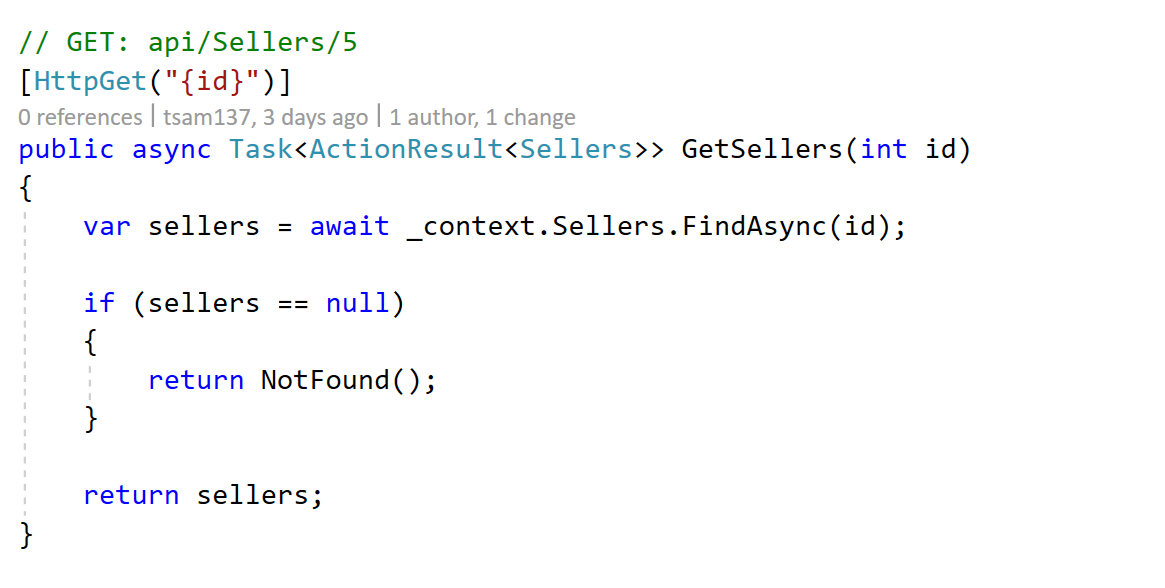
**Service**

**Sellers**

Get sellers: it gets the list of sellers in the database.



Get sellers/id: this is used to search for the sellers information by their id from the database if a null value is entered by the user a 404 code is shown.



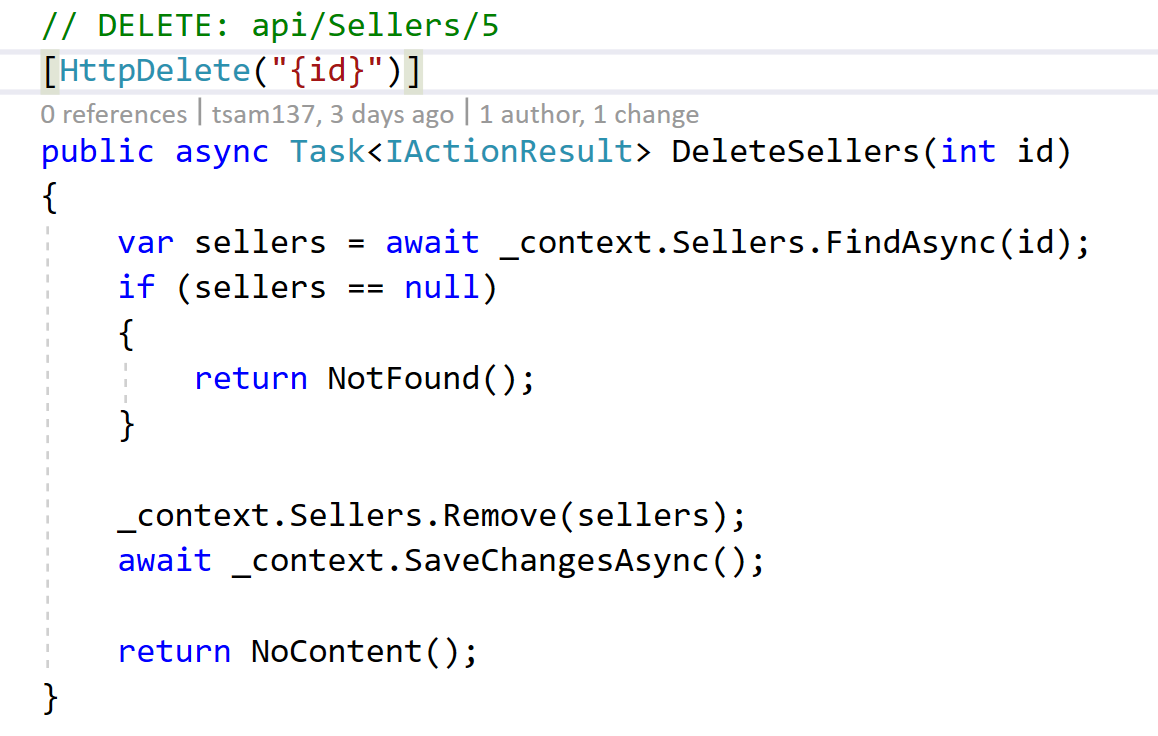
PutSellers : uses the seller id to allow the seller to update the information which is needed if their information is wrong or outdated. If there isn’t a seller with the id entered a 400 message. After the database it checks if the Seller exists if not it sends a 404.



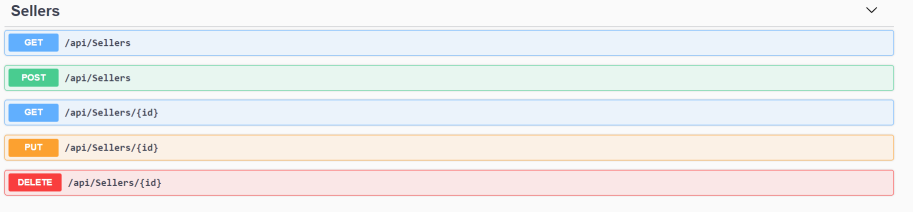
PostSellers: is used to create new sellers into the database by using addSeller. you enter the values of the new seller when you execute it, it then then saves the changes. once its saved to the db its returned as the id of the seller.



DeleteSellers: for this you input the id of the seller you would like to delete from the database if no id is entered a 404 message is returned. Once an id is removed the database is saved.

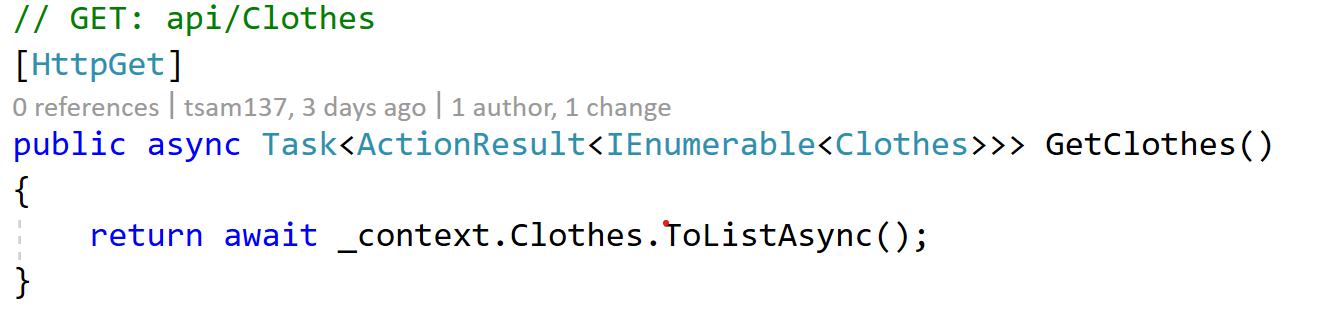


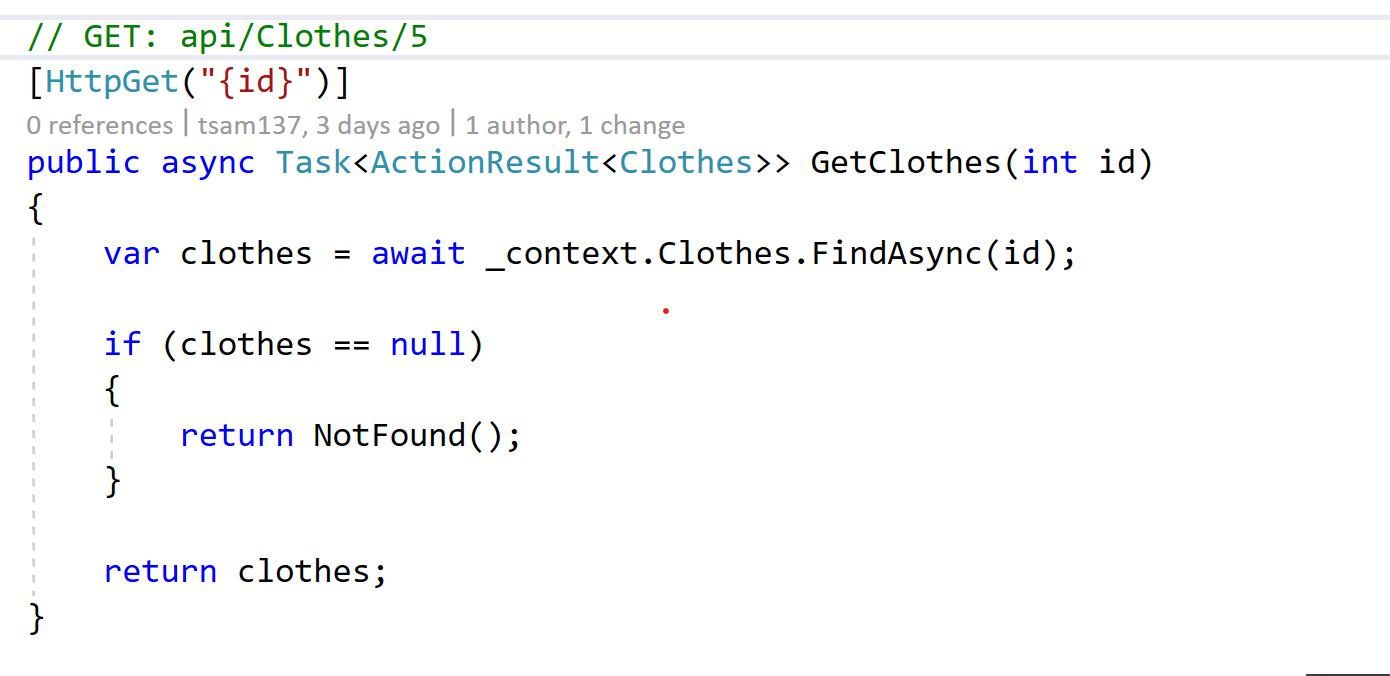
**Swagger API UI - Sellers**



**Clothes**

GetClothes: calls the database and returns the all the different clothes information.



GetClothes/id: this calls the database by the id that is used if you're looking for a certain piece of clothing. if there is no clothes at the id a 404 message is returned.

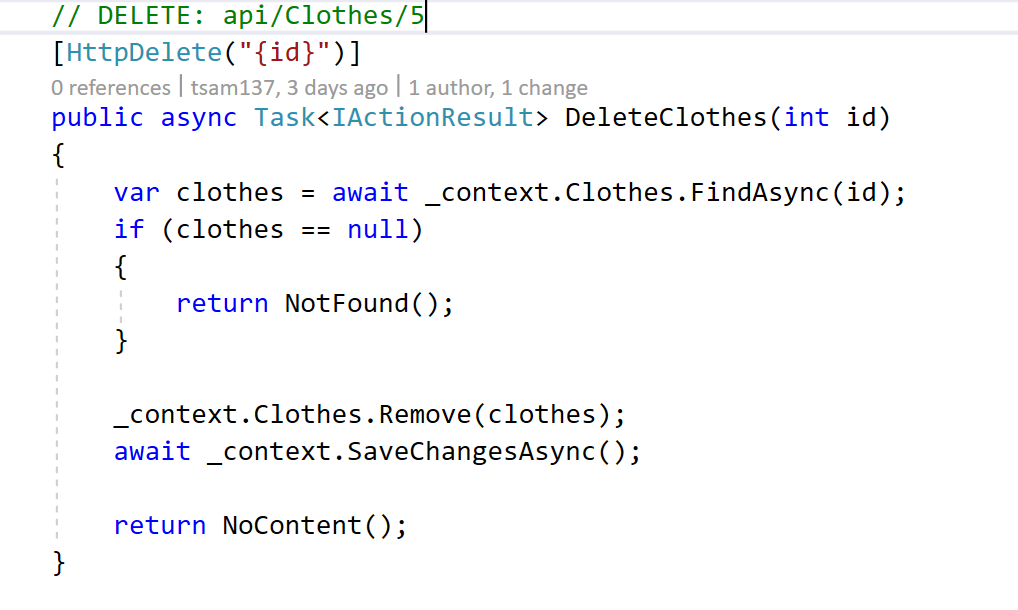
PutClothes: this call is made to update the clothes database using the clothes id if the id is outdated or incorrect information. If there aren't any clothes at this id a 400 message is returned. If after it saves to the database it checks if clothes exist if not a 404 message is returned.



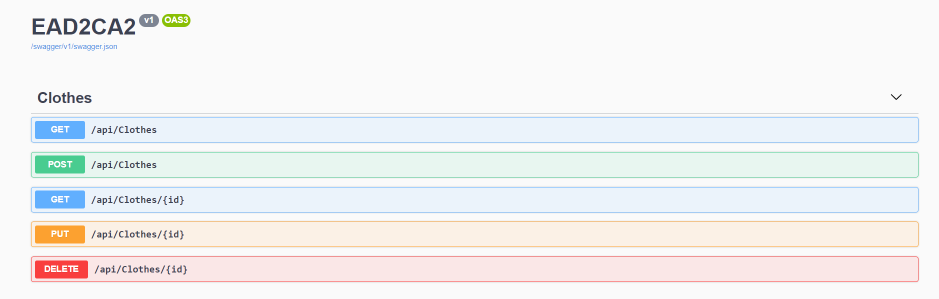
PostClothes: is used to create new clothes into the database by using addclothes. you enter the values of the new clothes when you execute it, it then then saves the changes. once its saved to the db its returned as the id of the clothes



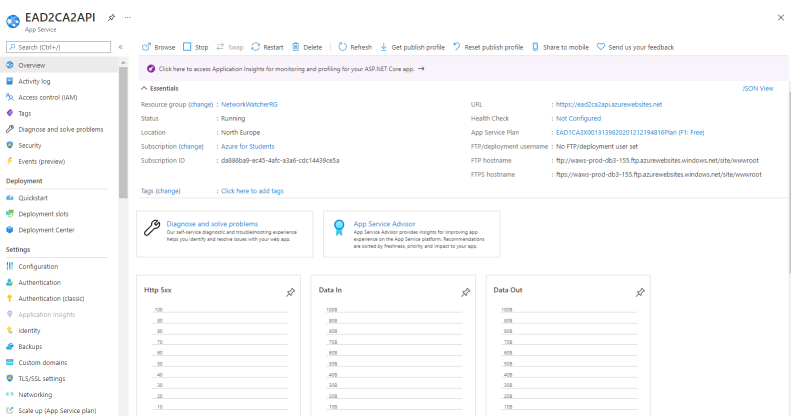
DeleteClothes: or this you input the id of the clothes you would like to delete from the database if no id is entered a 404 message is returned. Once an id is removed the db is saved.



**Swagger API UI - Clothes**

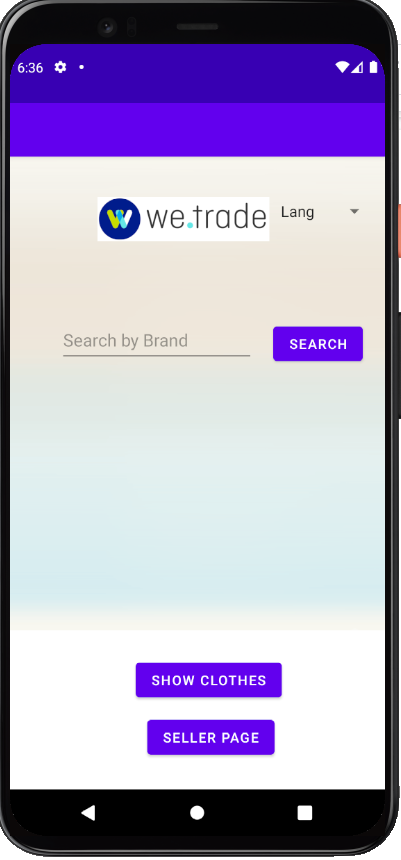


**Azure API Overview**

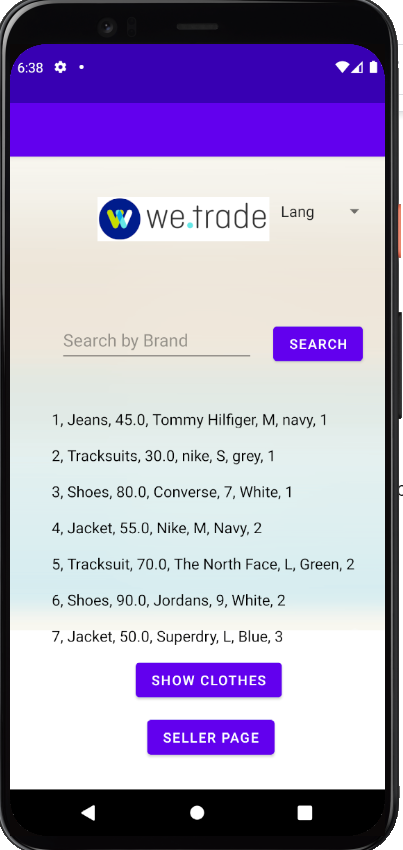


**App**

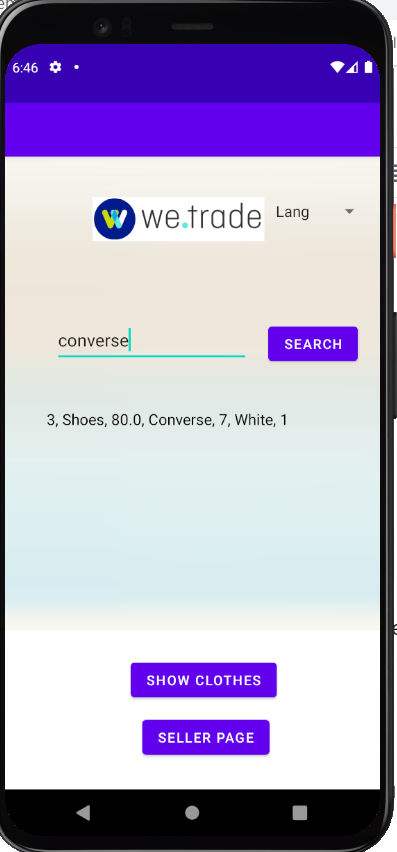
**Screenshots**

****

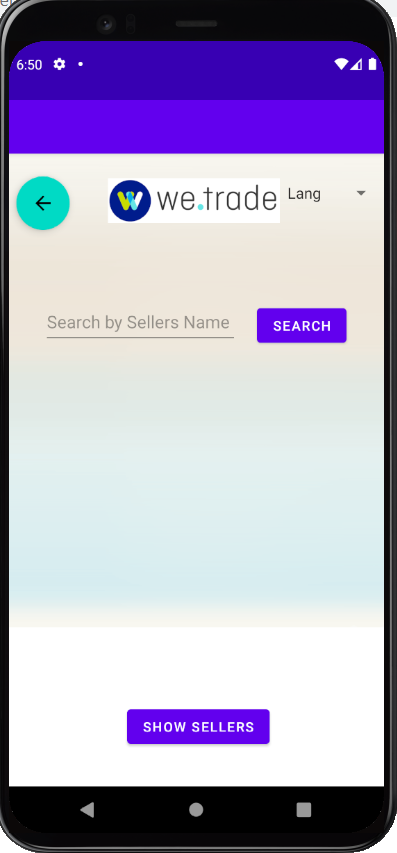
This is the first page that the user is greeted with when the app opens. Here you can see the logo and beside that there is a spinner(dropdown) that will allow you to change the language of the app. Under this you will see a search button that will allow you to search whatever the user enters into the search bar beside it. If you click Show clothes you will be greeted with the following page.



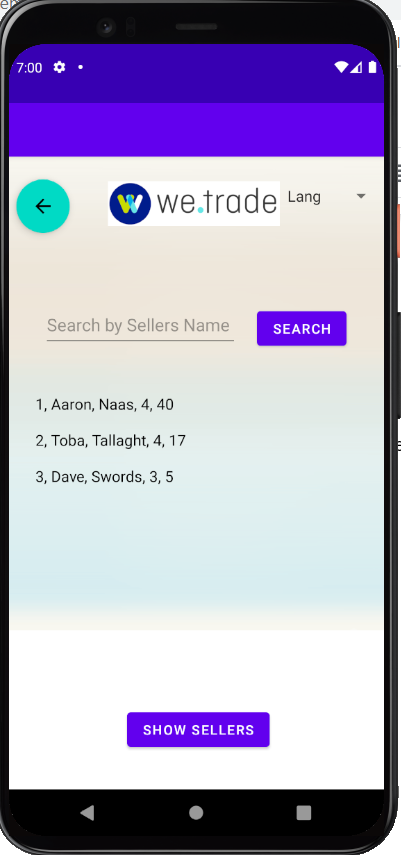
Here we can see the clothes that have been parsed from the URI web api link. If we click on the search bar and enter a name of the clothes or shoes then we can see the specific clothes we searched for as we can see below. I searched for converse and the converse item showed.

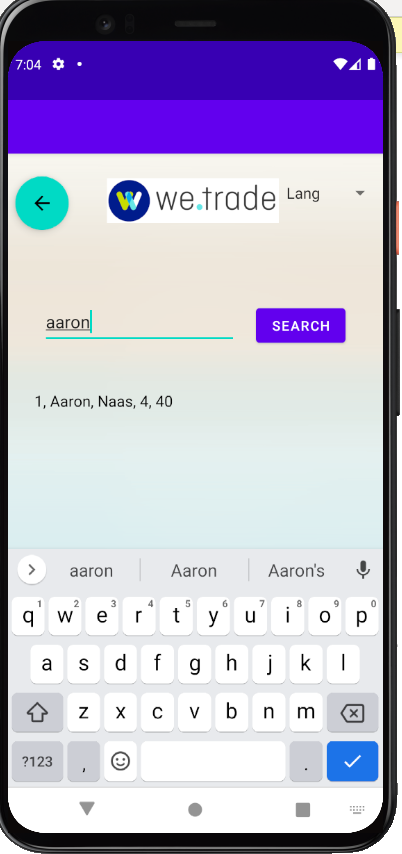


After this we can click on the seller page and we will be greeted with what we have below. This is the seller page and on this page we are able to go back to the clothes page. We can change the language of the app. We can search a seller by their name and we can get all the sellers.

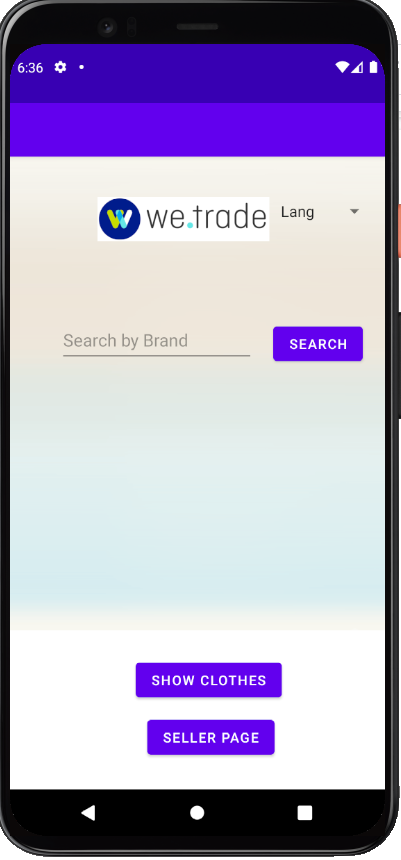


If you click on show sellers then you will be greeted with the following screen. Now you can see all the sellers in the database displayed in the app. If you search for a seller by name then you will be able to see only that seller.



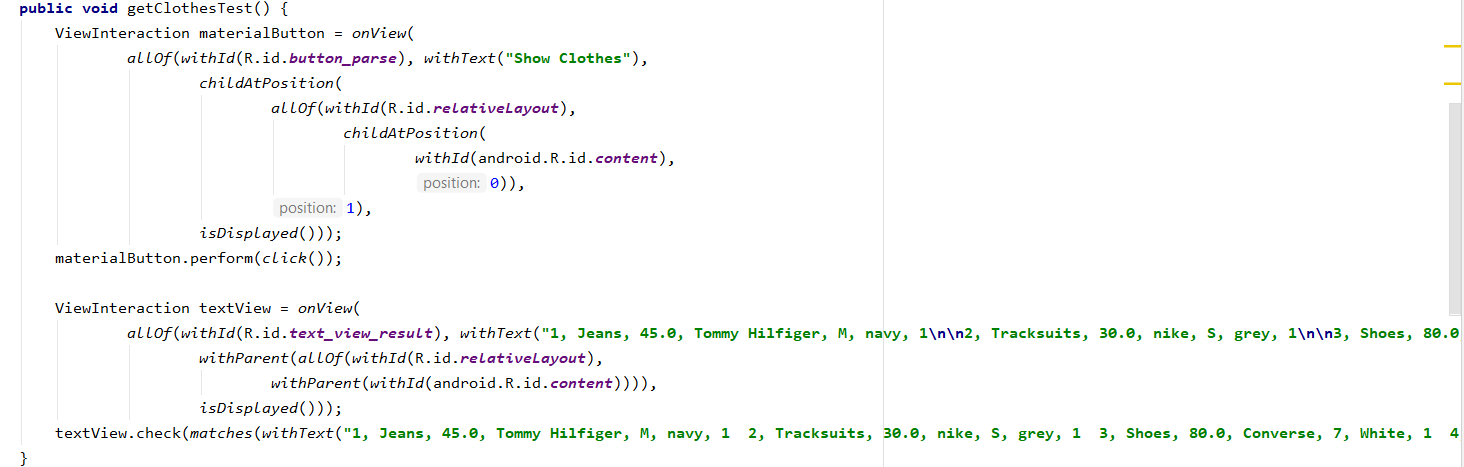
****

After clicking on search we only see this seller. If we press the back button in the top right hand corner, it brings us back to the clothes page as seen below.

****

**Testing Report - Expresso**

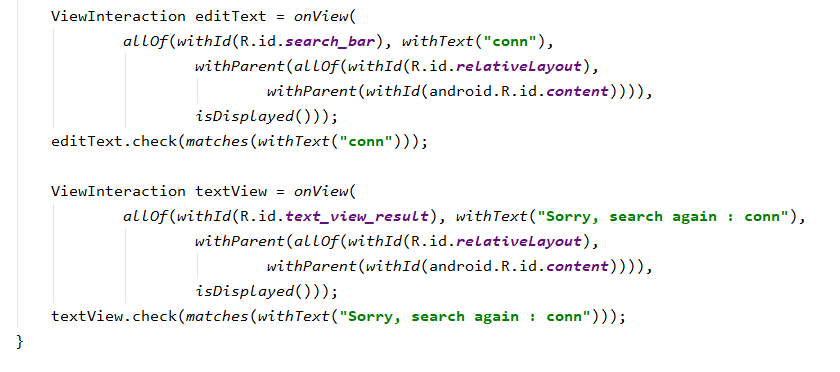
**Get Clothes Test** - Testing to see all sellers are printed when the Show Sellers button is clicked.

****

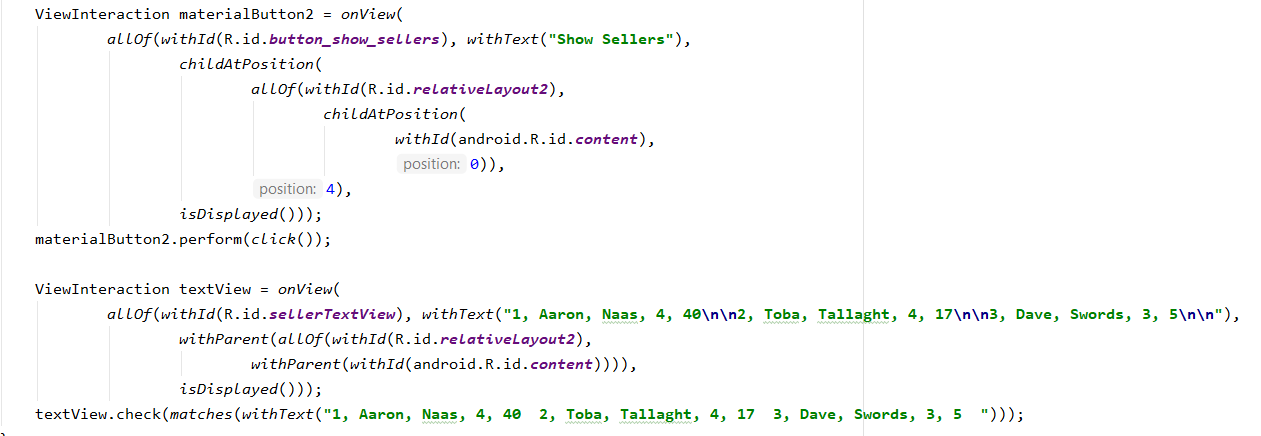
**Search Clothes Test** - Testing to see if what was searched for is what was shown.



**Search Wrong Clothes Test** - Testing to see if error messages appeared as wrong clothes were searched for.



**Get Seller Test** - Testing to see all sellers are printed when the Show Sellers button is clicked.

****

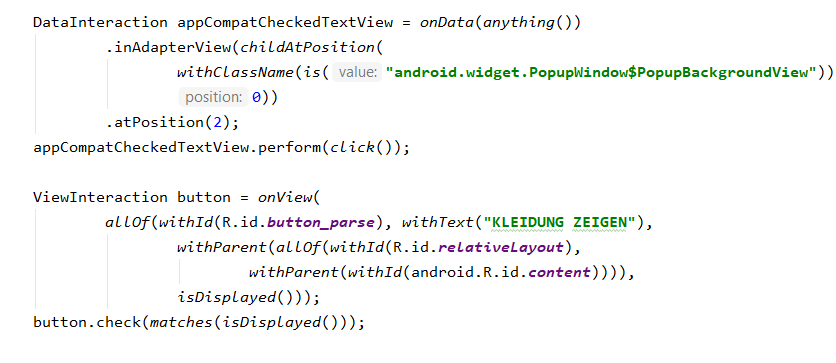
**Search Seller Test** - Testing to see if what was searched for is what was shown.



**Search Wrong Clothes Test** - Testing to see if error messages appeared as the wrong seller was searched for.



**Language Change Test** - Testing to see if what was searched for is what was shown.

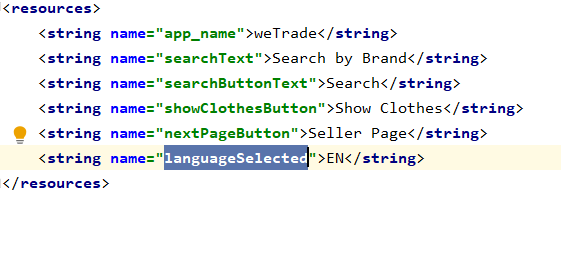


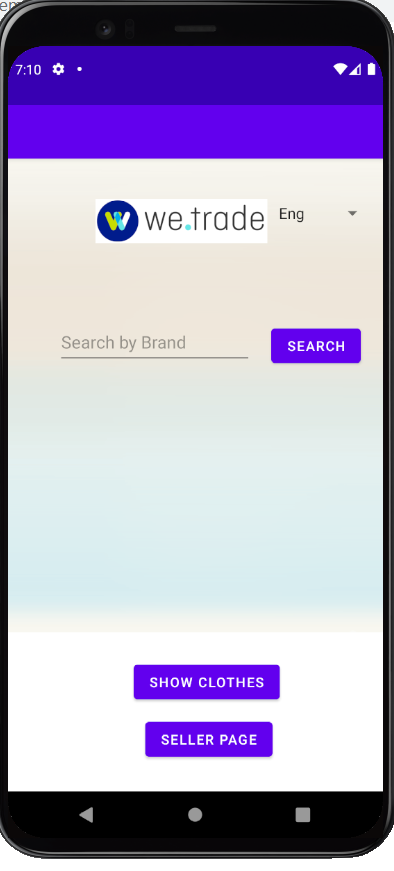
**Internationalisation**

The language we decided to add to this app is German. We wanted to add German as we felt it is a prominent language and many people speak it. To do this I added another strings.xml file that was more tailored towards the german language.

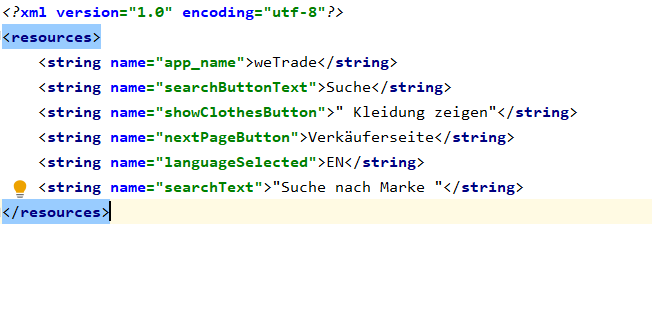
****

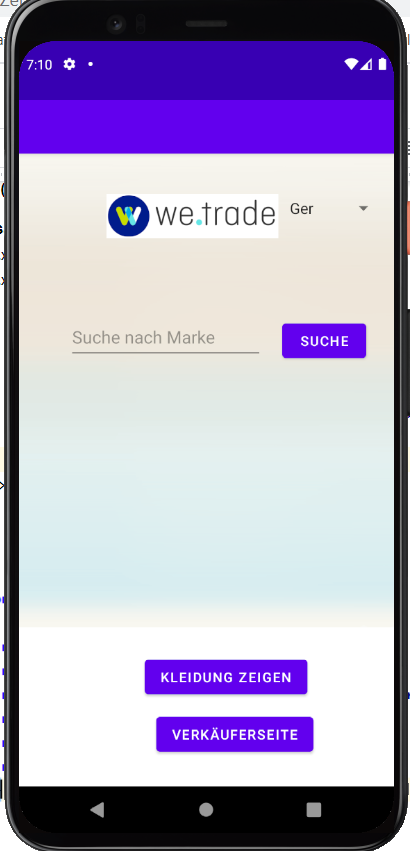
Below is the strings.xml file but the text is in English. These are the variables used in the app. These are displayed as English when the app first opens. And after the screenshot of the code we can see a screenshot of the screen when the language dropdown is set to English.

****

****

Below is the strings.xml file but the text is in German. These are the variables used in the app. These are displayed as German here when the user clicks on the German option in the dropdown menu to the right of the WeTrade logo. And after the screenshot of the code we can see a screenshot of the screen when the language dropdown is set to German.

****

****

**Miscellaneous – Code Quality Analysis**

For our linting tool we used the tool below “Language Support for Java(TM) by Red Hat”. We used this tool to make sure that as we typed there would be live reporting of parsing and compilation errors. This tool also helped us with Code completion, writing appropriate Code, Source actions and Refactoring. Some syntax is refactored and this tool helped us with them.

