TEESSIDE UNIVERSITY

SCHOOL OF COMPUTING

**SOFTWARE ARCHITECTURE (COM3041-N) INDIVIDUAL DOCUMENTATION**

Student Name: **Michael Watts (N3071956)**

**Work Packages**

At the start of the project architecture I had planned to cover the areas of relevant databases needed, caching of products and authentication. However, I came to realise that I may not have time at the end to run through authentication, the authentication process has been done as each team member has built their parts of the software.

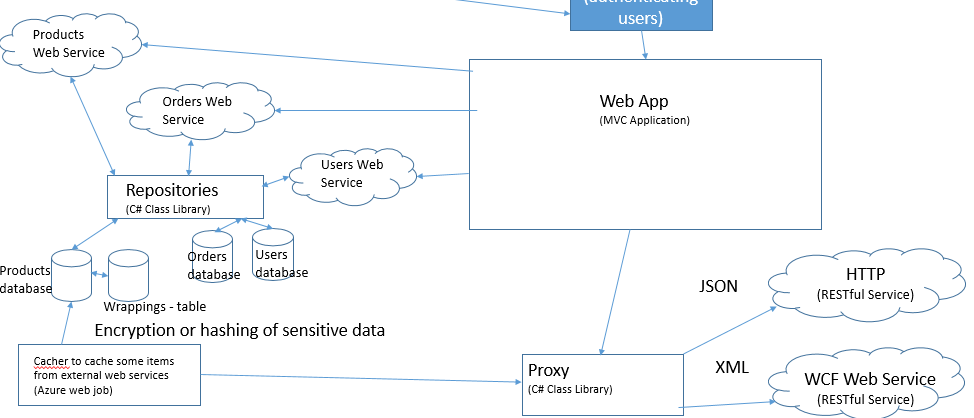
Originally it was planned to have a database for admin staff, however I came to realise that azure provided me with authentication using @tees.ac.uk accounts. This gave me the relevant security and meant that we did not need the extra database causing un needed stress on the application. This did have its limitations in a way that I had to be registered with @tees.ac.uk to have an account, in the future I would need to discuss and plan for change using other accounts.

The cache layer also had to be changed, the cache layer was going to take each product and a wrapping and store them in a database and update them locally every hour. However, I thought it may be more useful to create a web job program as a project that still stores the products and wrappings into a database but the program would be uploaded to azure that would run it every hour. This meant that less coding was done, but the job was smaller and meant that azure could handle this easily without any interruptions from the application.

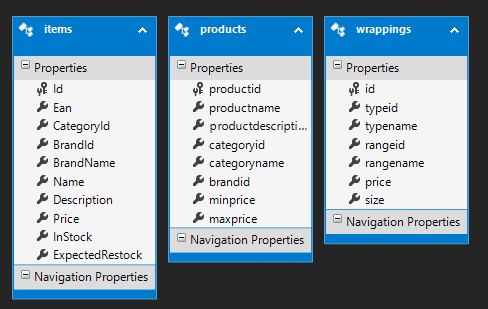
**Requirements Satisfied**

* Update any data from our suppliers at least once per hour.

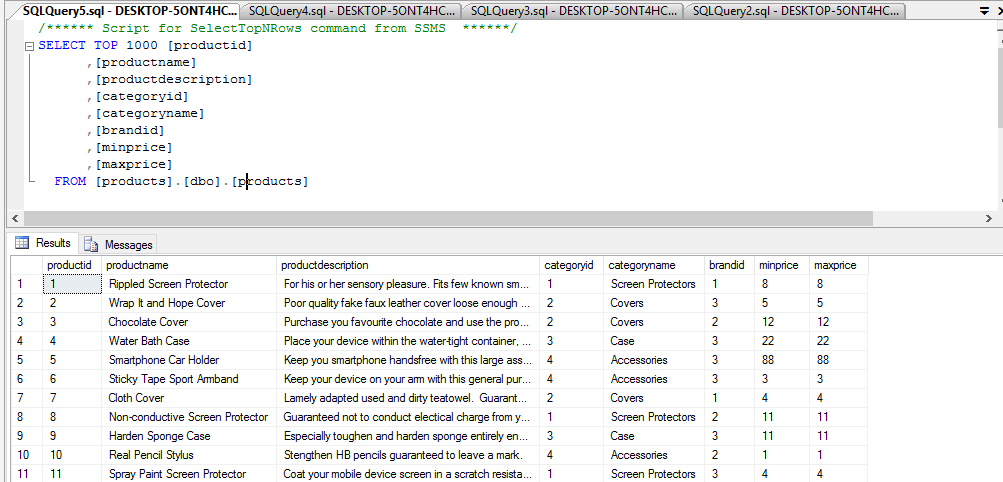
The requirement of this have been met by creating the cache layer by receiving products and wrappings from the service proxy of bazzasbazzar, khanskwikimart and undercutter stores. The cache web job stores the products and wrappings into a database and azure keeps it for an hour then automatically updates the services every hour. I have also handled the loss of data and data clashes by making sure that everything in the database is dropped before running the web job again populating new data received from the service proxy.



*Picture 1 – shows communication between cache and proxy*



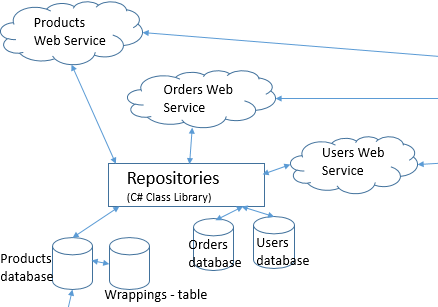
*Picture 2 – shows the view models created for cache layer & web services*

**

*Picture 3 – Shows data being stored from the web services.*

* Databases needed

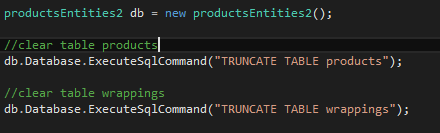
My other requirement was to make sure the databases where created for use of the cache project but also the databases needed for the repositories and web services. Firstly, I created four databases need for the web services to communicate to through the repositories. A products database was created to store products used by the web service but also had a table to store the wrappings. An orders table was used to store recent orders by the orders web service and a user’s table was created to store staff members or general customers.



*Picture 4 – Shows the communication between databases and web services through repos*

* Handling customer demand and duplicate entry’s

I came to realise that it was a good idea to clear the database before the cache program proceeded to store the data. If I didn’t clear the data before every time the job was run I would end up with duplicate entry’s which would cause the database to become full and strop the web job from running quickly each hour therefor handling the customers’ demands.

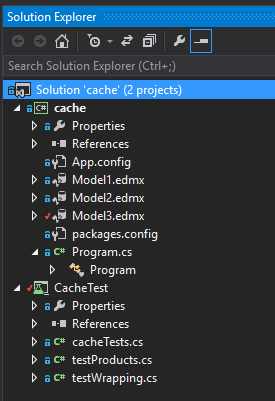


*Picture 5 – shows the table deleting duplicate entry’s*

**Software Development Frameworks and Tools Used**

* Visual studio & entity Framework

Visual studio was used to create my web job as a project. This allowed me to create models of the database created and allowed me to gather the variables and data that needed to be stored and meant that I had access to it all in one place. Visual studio was also used to create the testing harness for the cache program.



*Picture 6 – Shows the solution explorer from visual studio*

* Sql server management studio

Sql server management was used to create local instances of the databases needed for our application to work and hold valid data. This helped us to store data locally first which helped with load testing and front end testing.

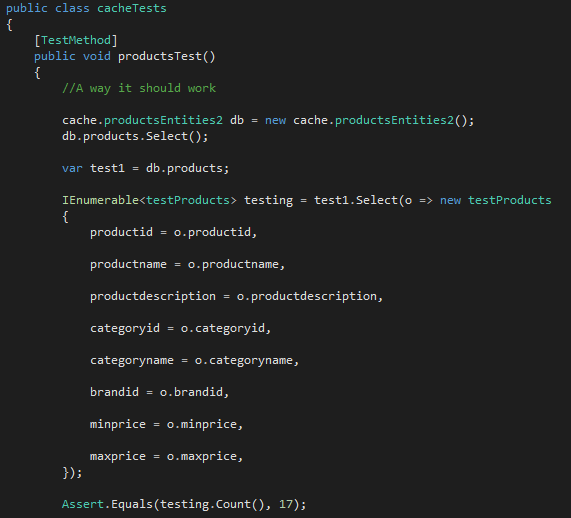
* Azure

I had planned to use azure web jobs to run my cache program once an hour to make sure that the data stored was valid. However, I ran into a problem when it came to uploading my web job as I had run out of credit. The orders and user tables were created in the beginning therefor I could upload them to azure without any problems.

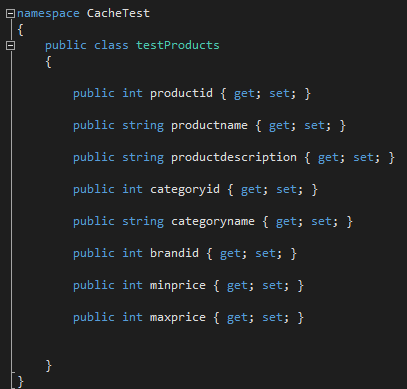
**Testing Performed**

* Products & wrappings testing

I created a representation of the database used inside the cache program that is used to store my products that need to be cached. I then created a list of objects from the database that where already stored I then make sure that the correct number of products have been stored e.g. There should have been 17 products stored to be a correct test. This is a basic test to make sure that there is data being written to the database.

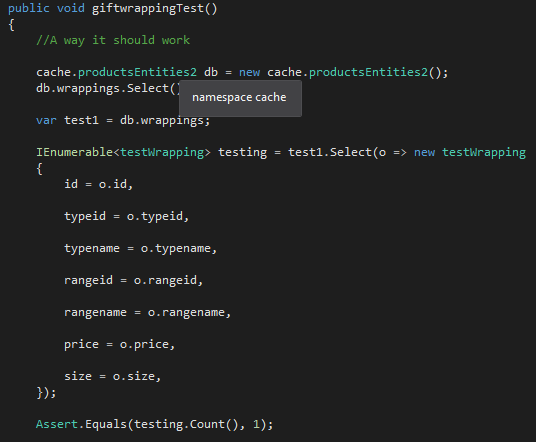


*Picture 7 – Shows the test class for products*

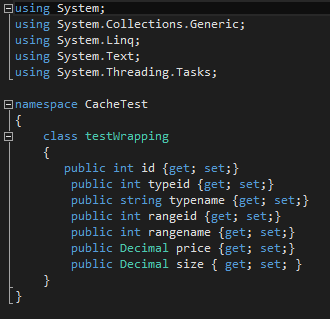
**

*Picture 8 – Shows the values from the model needed to test*

The same testing harness was used to test the amount of wrappings that had been stored inside the database were the correct amount.



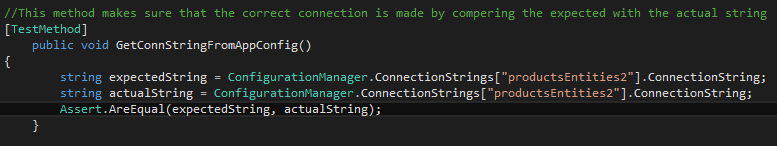
*Picture 9 – Shows the test method for wrappings*

**

*Picture 10 – Shows the values the test needs for wrappings*

* Testing the connection string

This test allowed me to test the connection string for the database. This is done by comparing the actual database connection string with the expected string. It gets the database connection string from the app.config file. The test tells me it passed and passes the correct string.



*Picture 11 – shows the database connection string test*

**Deployment Process Proposed**

After my research into using an azure web job to run the cache, I planned to use Azure to run the job every hour that would suit the purpose in storing the data needed to handle customer demand. Unfortunate, I did not manage to get the web job uploaded to azure due to running out of credit. However, the database for the orders and users have been uploaded to azure as I found that this needed to be handled first for my group to move forwards.

**Research & Problems that occurred**

Originally, I was going to create a web job that took each individual product and wrapping and created as local object and saved it into the database and it kept it therefor an hour. However, with a little research I found that creating a web job that stored the products and wrappings into a database that would be hosted on azure could in fact be updated every hour and would have saved the project on load balancing and would speed up customer demand.

Unfortunately, Dan did not manage to get the web proxy working for BazzasBazzar, therefor I could not get the products to store to the database to cache them for an hour. I have left the code commented out for illustration purposes on how I would handle it if it was available.

**Installation**

* Databases

The database need to be included into the root destination of MySQL. Simply right clicking and attaching the database is required.

* Running the project

The program cache is executable as a web job but not deployed. A cmd line will appear quick then close, it will show an exit code 0 (0x0) this means it exited with 0 errors. After the program has been executed data will appear inside the database products and wrappings.

**Peer Marks**

Matthew Frost (L1426439) 5

Dan Bell (N3042486) 4

Michael Watts (N3071956) 3

Jack Wilkinson (P4063103) 3