Advanced Java (IT351-1904A-01)

Client – Server Application

David Blonn

November 5, 2019

C**ontents**

[Pseudocode for Client - Server 4](#_Toc23863587)

[Class Diagram 5](#_Toc23863588)

[Sequence Diagram 6](#_Toc23863589)

[Multithreaded Screenshots 7](#_Toc23863590)

[MySQL CRUD Through JDBC 9](#_Toc23863591)

[Create 10](#_Toc23863592)

[Update 11](#_Toc23863593)

[Delete 12](#_Toc23863594)

[Read 13](#_Toc23863595)

[Web Application Design 14](#_Toc23863596)

[Web-based Application User’s Manual 17](#_Toc23863597)

[Test Cases 19](#_Toc23863598)

[SQL Script to Create and Populate Tables 19](#_Toc23863599)

[Read Records 20](#_Toc23863600)

[Create A Record 21](#_Toc23863601)

[Update A Record 22](#_Toc23863602)

[Delete A Record 23](#_Toc23863603)

[Multiple Client Requests 23](#_Toc23863604)

[Web-Application Discussion 24](#_Toc23863605)

[Examples & Benefits of Web Services 24](#_Toc23863606)

[Troubleshooting & Consuming Web Services 25](#_Toc23863607)

[Technologies & Persistence for Web Services 25](#_Toc23863608)

[References 27](#_Toc23863609)

# Pseudocode for Client - Server

The connection from the client application to the server application will use Transmission Control Protocol (TCP). Utilizing the URL of the server, the client application will send a request to establish a connection over the internet. In Java this is achieved by using Sockets. Once the connection has been made an ObjectOutputStream or similar buffering tool can be used to send data objects from the server to the client and vis versa.

Pseudocode for Client Server interaction:

Client sends connection request to Server

Server accepts connection

Client sends request for product data to Server

Server receives request for product data

Server reads product data from file

Server sends product data to Client

Client displays product data to user

Client sends request for customer data to Server

Server receives request for customer data

Server reads customer data from file

Server sends customer data to Client

Client displays customer data to user

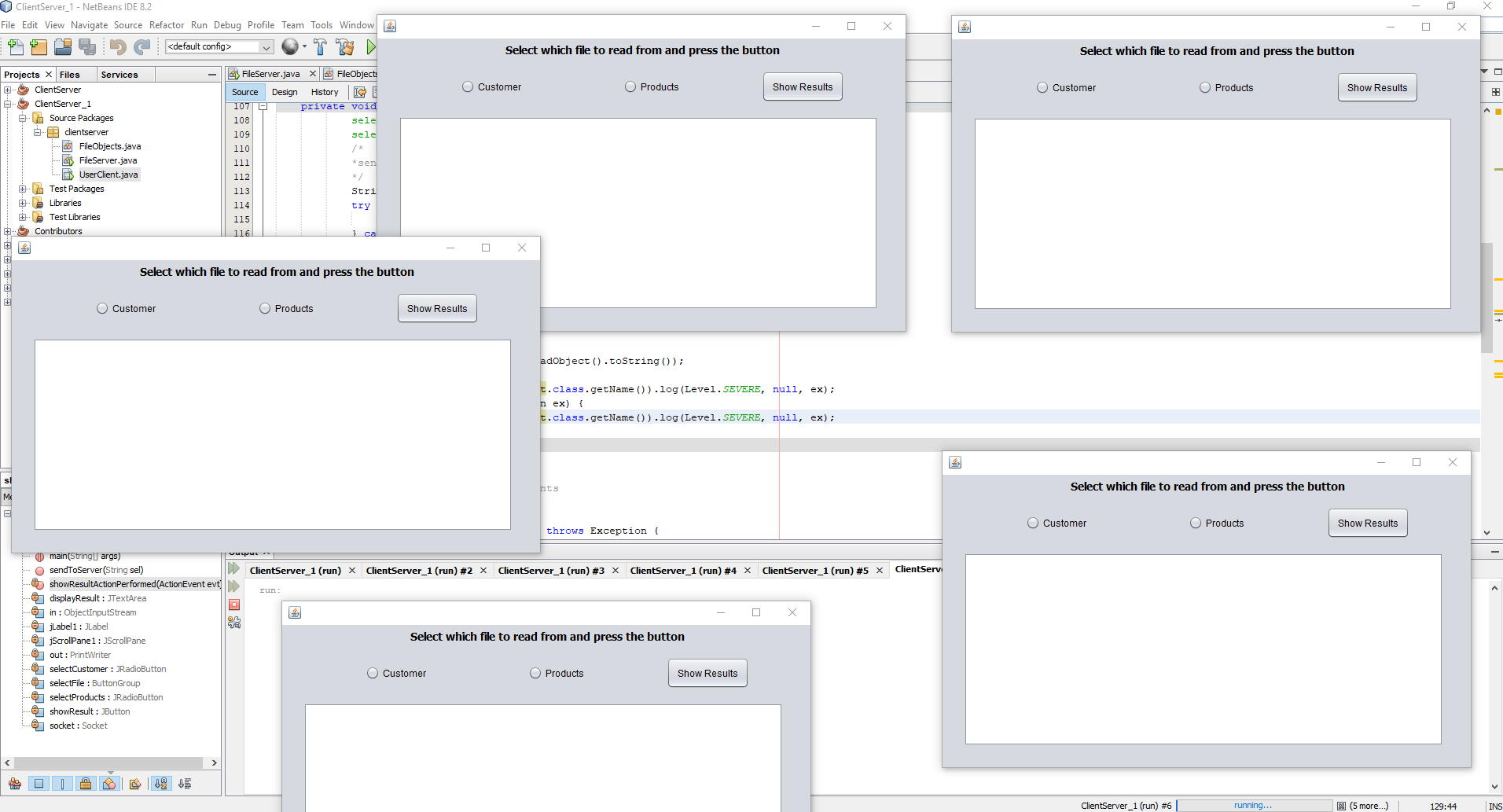
# Class Diagram



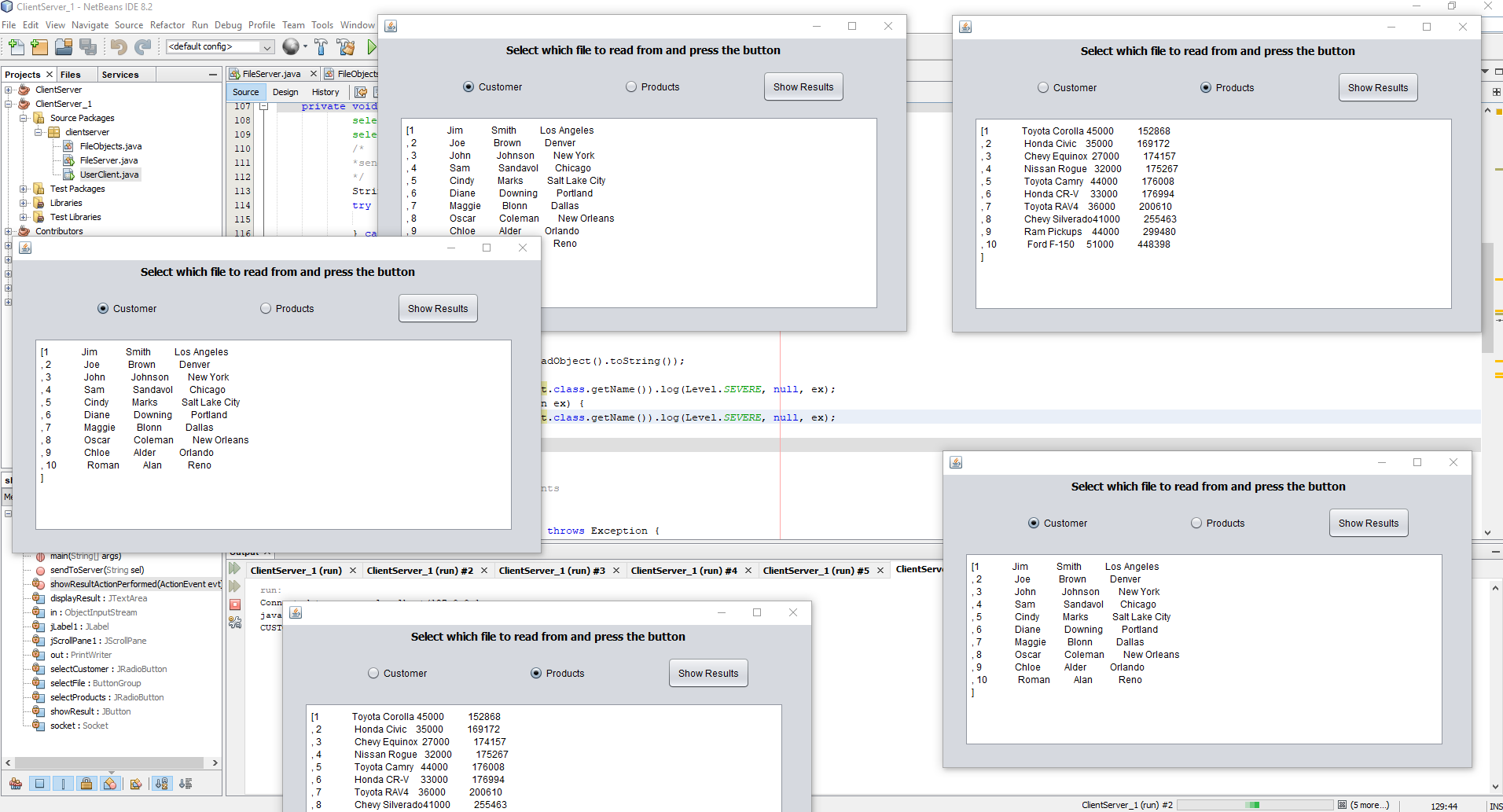
# Sequence Diagram



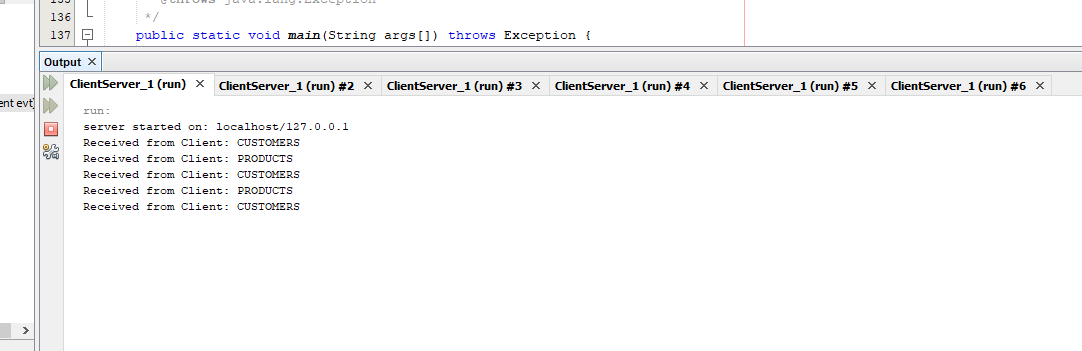
# Multithreaded Screenshots



1. The server is running, and 5 clients have been opened to connect to it.



1. Each client has requested a certain file from the server and received the data from the file back, for display



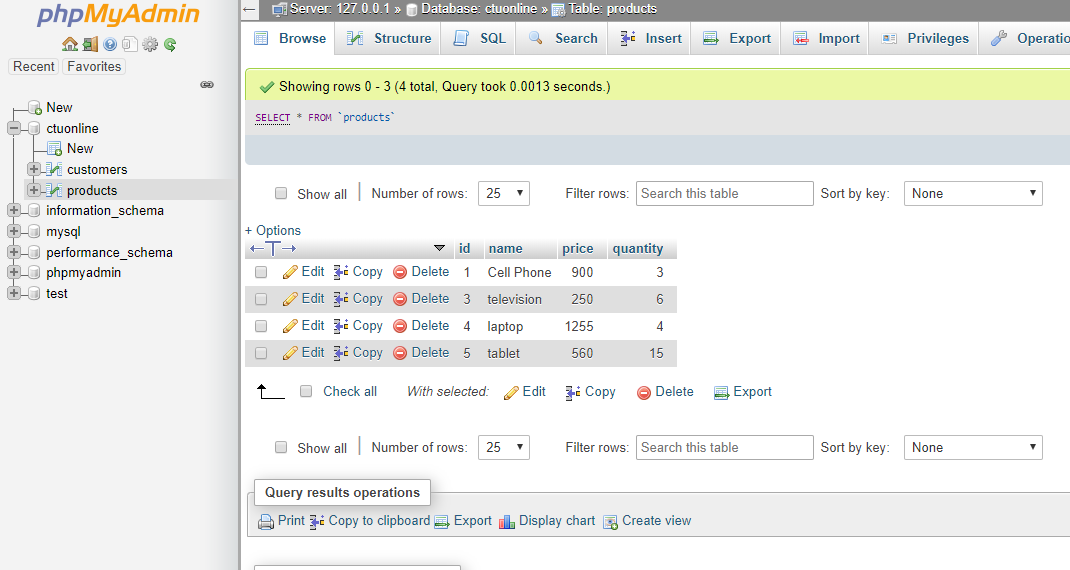
1. The server has logged the data that it sent to each of the clients connected to it

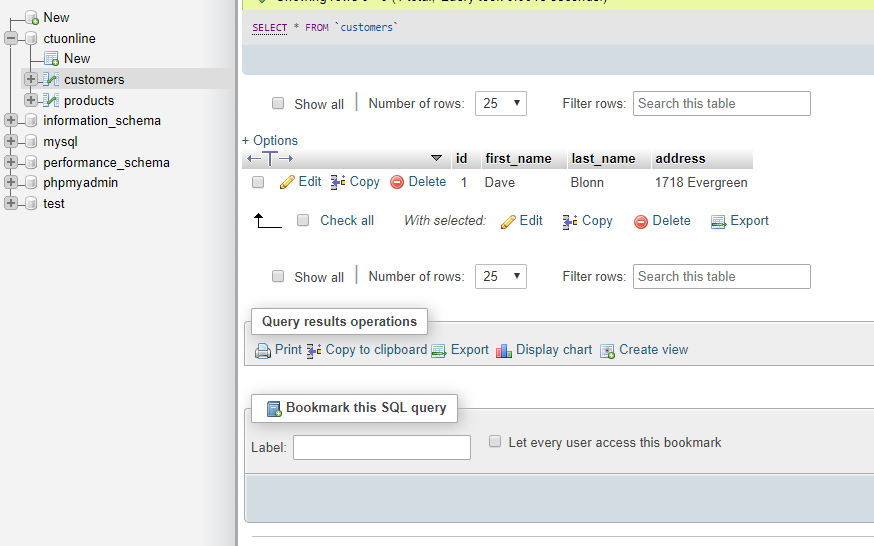


1. Each client has logged the port the PrintWriter sent its request on, and the type of file it requested information from

**MySQL CRUD Through JDBC**

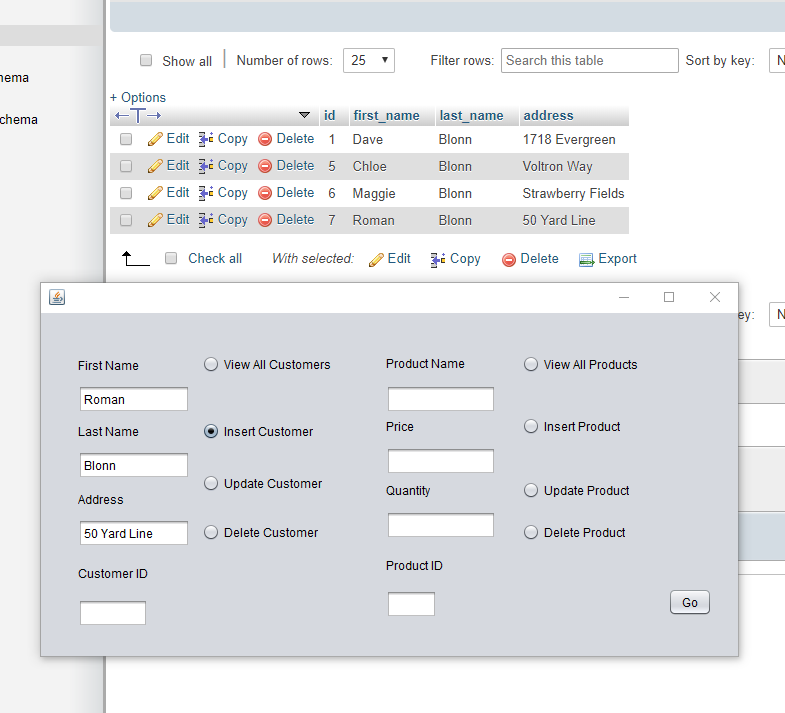
The first 2 screenshots show the beginning state of the relational database before the desktop application is opened. It shows that the Products table has 4 rows, and the Customers table has 1 row.

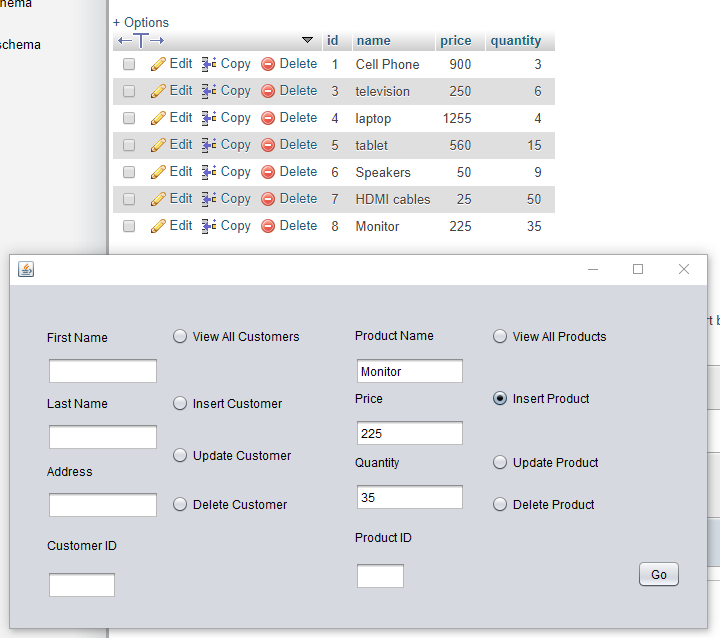
****

****

**Create**

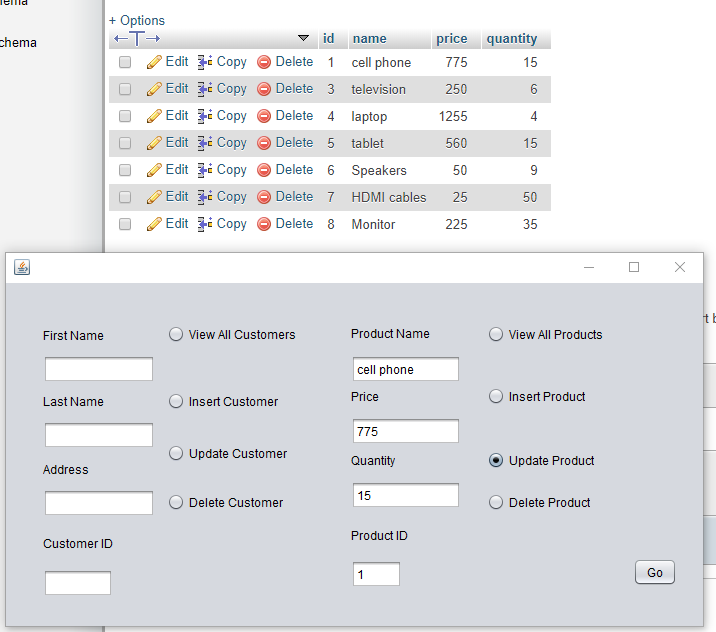
The next 2 screenshots show the desktop applications GUI screen and the functionality it offers to create records in the database. After using the insert functionality a few times, the added rows in the tables can be seen.

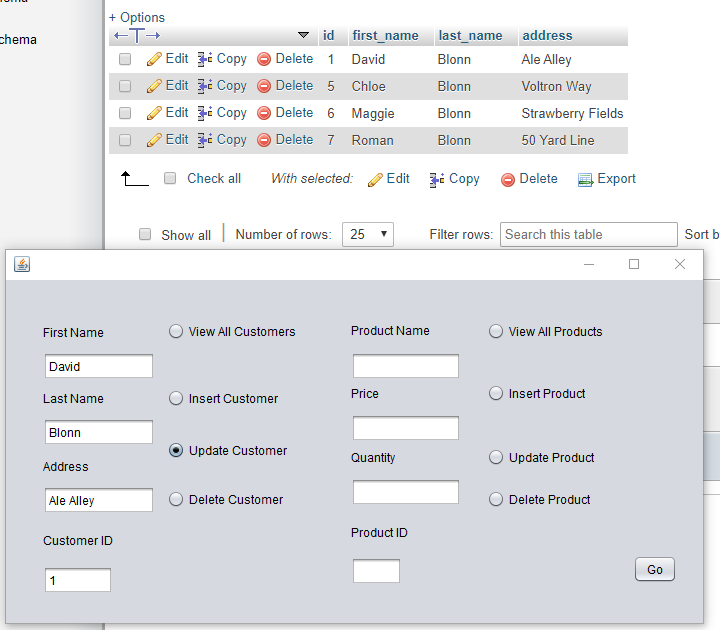




**Update**

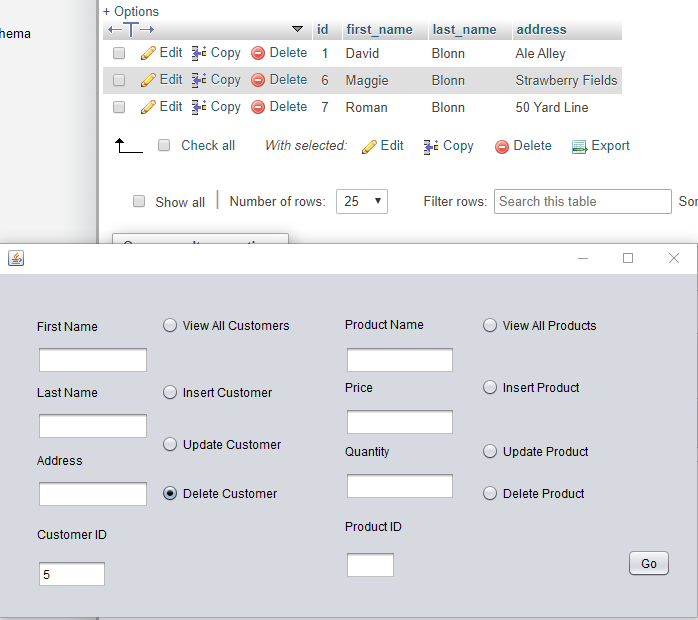
Though the ID’s of both tables are created and auto-incremented by the RDBMS as a primary key in each table, JDBC allows us to access the records using the ID’s. By providing the existing id of the Product or Customer, the database can update the record. The next 2 screenshots show that the product record for cell phones and the customer record for Dave Blonn have been updated.

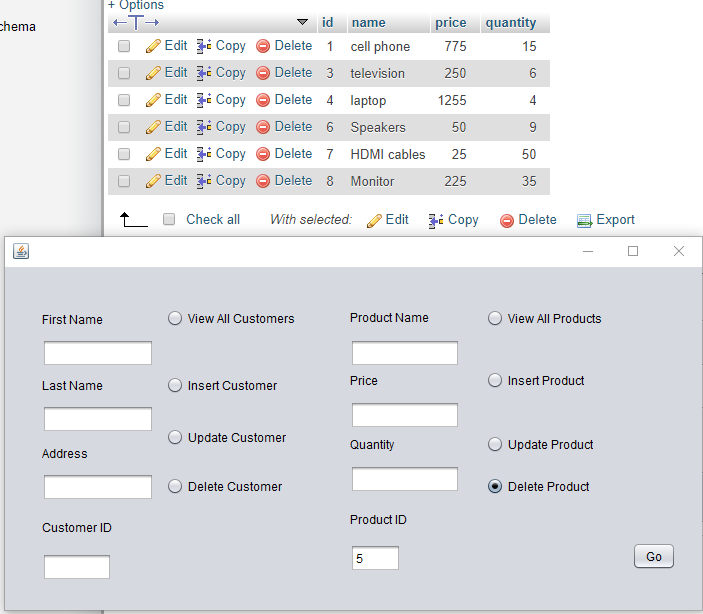




**Delete**

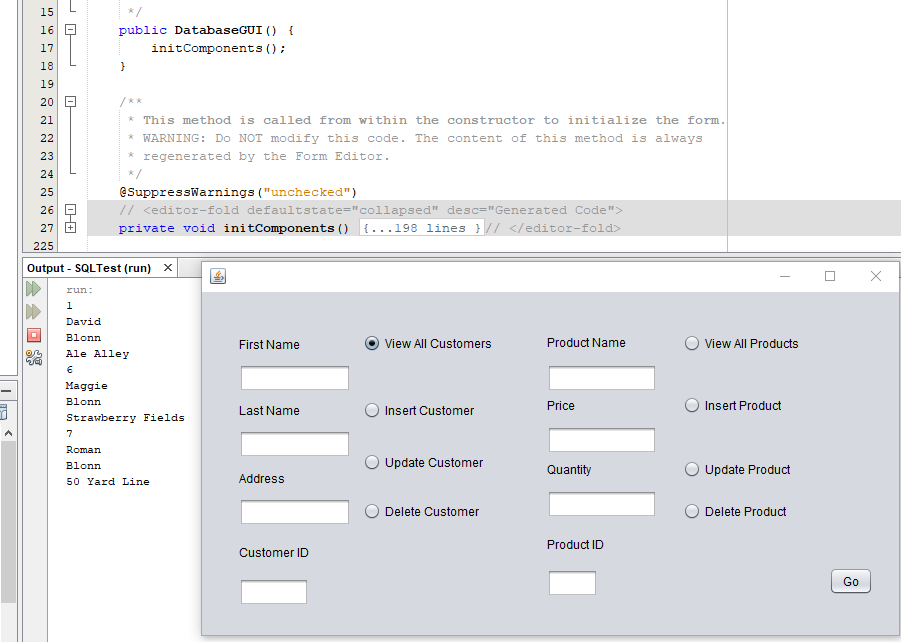
The ID’s can also be used to delete records from the tables. The next 2 screenshots show that record number 5 has been deleted from each table.

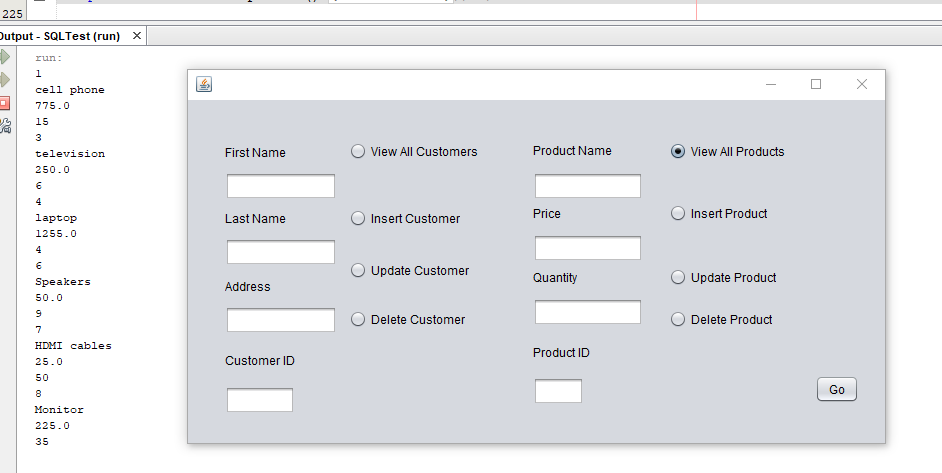




**Read**

Finally, the application allows you to read all of the records that are contained within a table. The following are screenshots of the results shown as a system print line





**Web Application Design**

A screenshot of a cell phone

Description automatically generated

The web application to support access to the database will be built in a multi-tier architecture that utilizes the JavaServer Faces (JSF) Model-View-Controller (MVC) framework. JSF’s MVC framework makes use of many other Java APIs that extend its functionality to include just about everything needed to deploy web-based applications. Enterprise Java Beans (EJB) class reinforces encapsulation for enterprise applications and can control the lifespan of objects throughout a user’s session. The Persistence class provides an EntityManager for objects that interact with the database.

The Glassfish server is a robust server provided with the JDK. It was developed for JavaEE applications. It works with Persistence, WebSockets, EJB, and other Java API’s to make enterprise application development easier. The server will process the request and send/receive updates to the database.

A close up of a map

Description automatically generated

The Facelets of the JSF bring a lot of technologies together to create the Controller in the MVC framework. This is achieved by embedding dynamic Java commands inside of the static XML code. CSS handles the structure. The Facelets are written using tags, which reference tag libraries to provide the functionality.

Other technologies:

Java Server Pages (JSP): JSP’s combine HTML, XHTML, and XML code with embedded Java code to create dynamic web pages. JSP’s are an integral part of JavaEE; capable of using all the resources enterprise applications may need. JSP’s could be integrated into the JSF used in the current architecture if additional functionality was necessary. However, they are not capable of handling some of the web-based functions that are built into JSF.

Apache Tomcat: Tomcat is a web server. It is popular for its low-overhead and ease of implementation. For this project, I have chosen to remain with the more robust Glassfish server that provides many of the EJB, JPA, and servlet tools necessary (and already integrated into) JavaEE.

**Web-based Application User’s Manual**

1. In MySQL, create a database named “ctuonline.” The username needs to be set to “ctuonline.” The password needs to be set to “student.”
2. In Netbeans, navigate to the Services tab, open up the Database Explorer, and ensure that Netbeans is connected to the database.



When connected, open up the icon that shows the database connection for ctuonline. Right click the Tables folder and choose Execute Command.

1. Netbeans will open a blank SQL Editor form. With in the SQL Editor, write the following SQL Script:

CREATE TABLE `products` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`prodName` varchar(500) NOT NULL,

`prodPrice` float NOT NULL,

`prodQuantity` int(11) NOT NULL,

CONSTRAINT products\_id PRIMARY KEY (id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `customers` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`firstName` varchar(500) NOT NULL,

`lastName` varchar(500) NOT NULL,

`address` varchar(800) NOT NULL,

CONSTRAINT customers\_id PRIMARY KEY (id)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO products (prodName, prodPrice, prodQuantity)

VALUES ('tablet', '115.50', '45'),

('television', '249.99', '35'),

('laptop', '1265.87', '22'),

('monitor', '156.99', '16');

INSERT INTO customers (firstName, lastName, address)

VALUES ('David', 'Blonn', '170 Knollwood'),

('Oscar', 'Blonn', '2nd Bedroom'),

('Chloe', 'Blonn', 'Track Meet'),

('Roman', 'Blonn', 'Wrestling Mat');

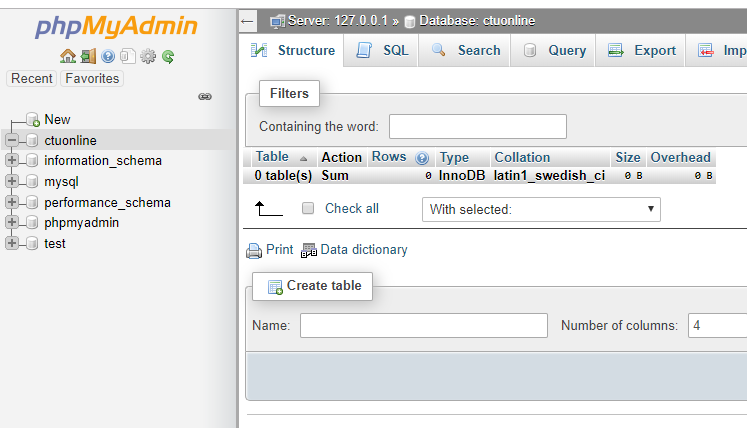
Right-click the SQL Editor form and choose Run File. This will create the Products and Customers tables in the database.

1. Return to the Projects tab in Netbeans and right-click the OrderWebApp icon. Choose Run. This will begin execution of the program.
2. A new web page will appear in your default web browser. This is the index page of the web application. You can choose to view the customers or products tables.
3. After choosing a table to view, you will be presented with the option to View, Edit, or Destroy any of the individual records already in the table, or you can Create New Record. Use the Index link to return to the index page.

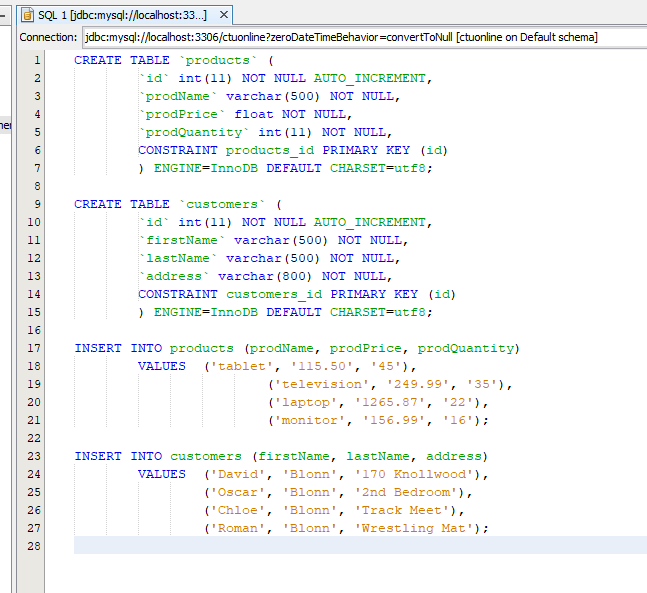
# Test Cases

The Web Application will fulfill the necessary create, read, update, and delete (CRUD) functionality for managing the database. Each CRUD test on the system has been documented. The bulk management of the database will be handled through SQL scripts. A test script with successful execution has been included in the tests. The system will also be able to handle multi-threading operations. The database will be able to receive requests from one client and make synchronous changes to the database that are available to any other connected clients.

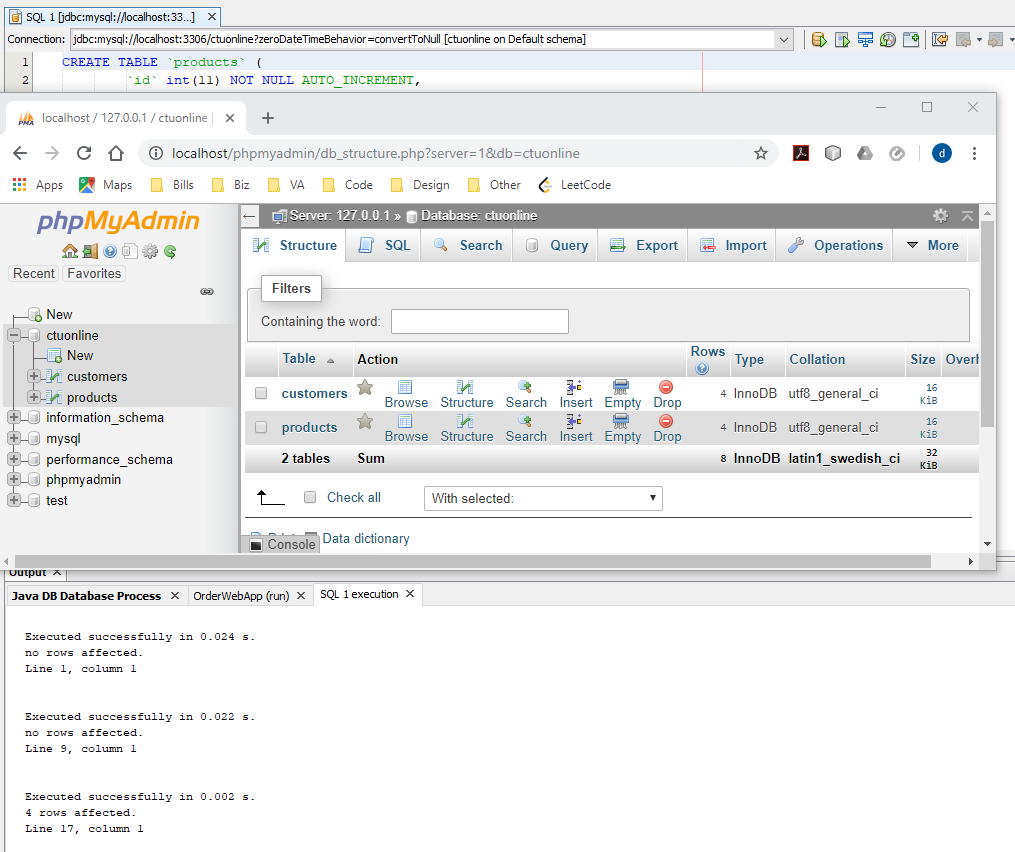
## SQL Script to Create and Populate Tables



MySQL Database ctuonline is shown with no data in it

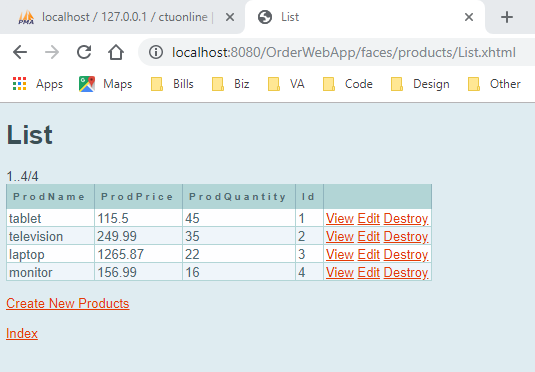


SQL script is executed on the server from Netbeans



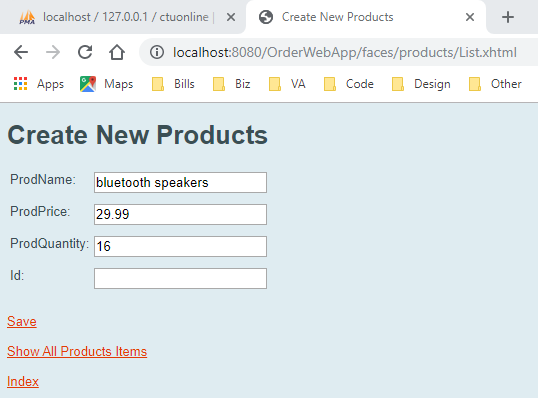
MySQL can now be seen populated with the correct tables, and the server print out shows the affected rows from the SQL script.

## Read Records

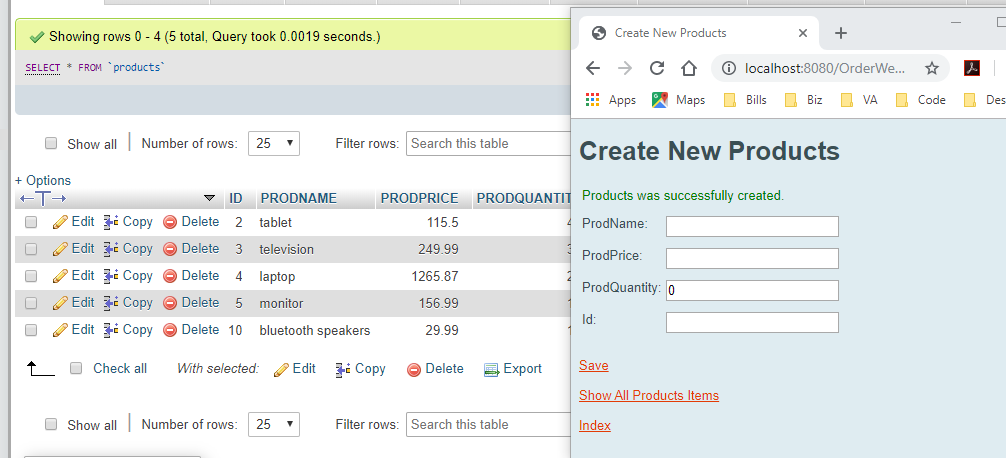


Current records of Products as read from MySQL and displayed to the client side browser.

## Create A Record

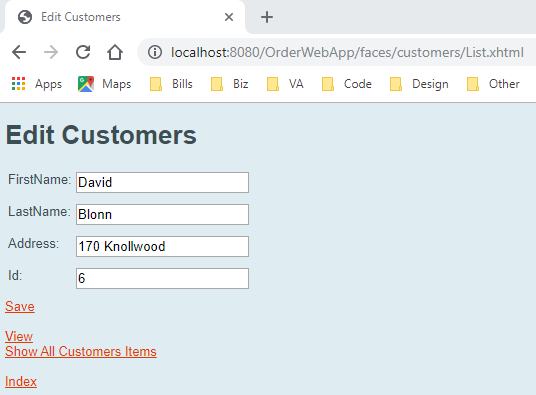


Using the Java Server Faces client to create a new Product record

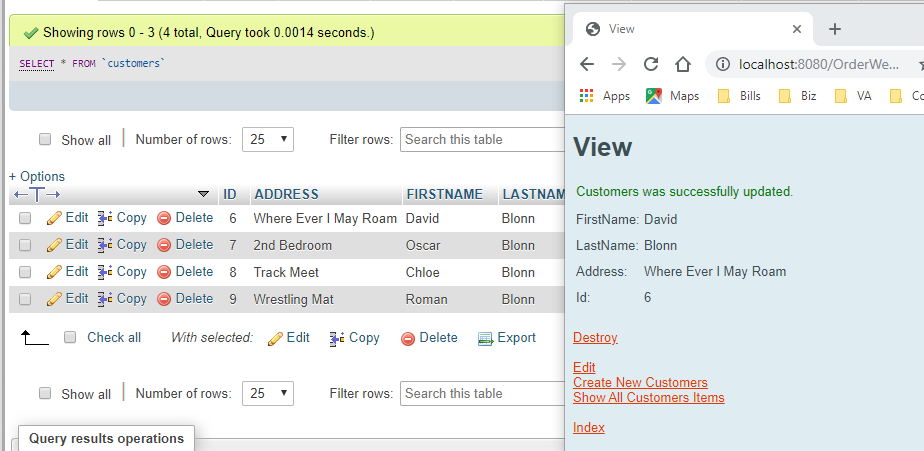


The browser displays a record was created successfully, and the new record can be seen in MySQL

## Update A Record

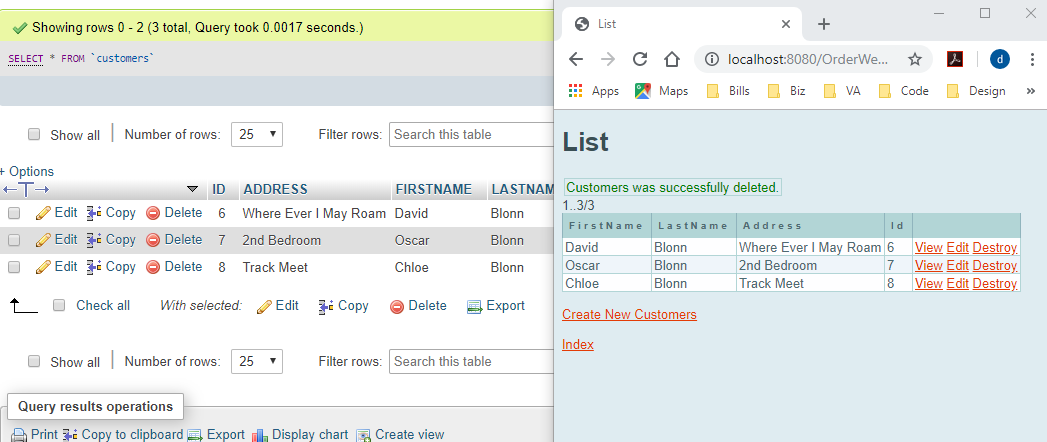


Navigated to the page to edit a customers information



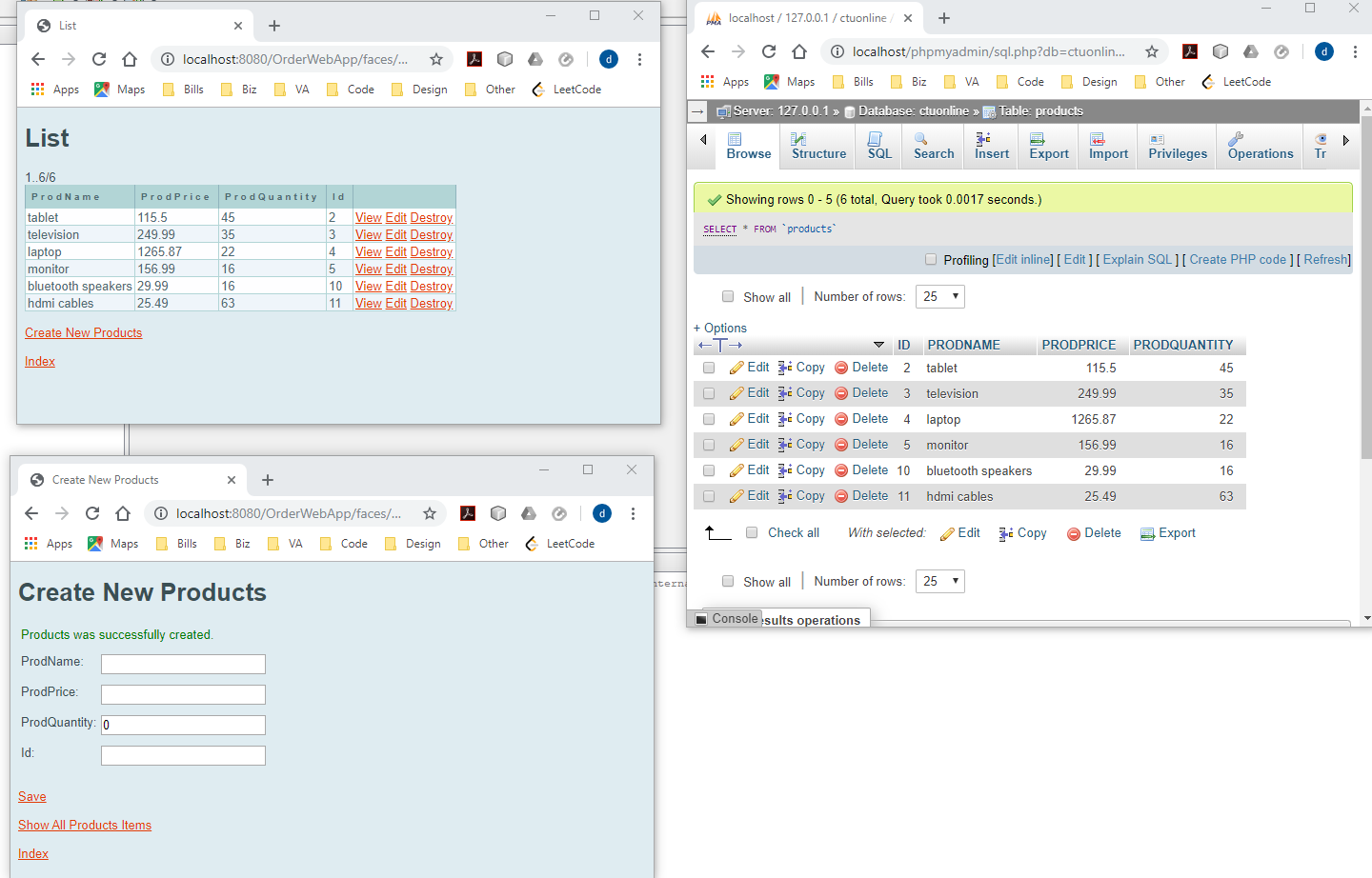
The browser displays the results of the edit, and they are seen in MySQL

## Delete A Record



Choosing the Destroy link from the browser executes the deletion of the record.

## Multiple Client Requests



Multiple Clients have connected to the database. In one browser a create operation was performed, and the result can be seen in the list of products on the second browser.

# Web-Application Discussion

## Examples & Benefits of Web Services

In the real world, web services have become an indispensable part of business and personal life. Web services have leveraged a standardization in the communication methods between the enormous variety of programming languages found in client and server programs. Amazon Web Services (AWS) is a subsidiary of Amazon that has specialized in providing web services to customers. By placing your program on an AWS server, you can utilize the server-side functionality they offer, as well as receive requests from clients for computing. AWS is an architectural system that is designed to deploy applications on the web by providing a robust set features for web services. More generally, the vast number of shopping applications that are available for mobile devices utilize web services to make buying, paying, and shipping their products easier for consumers. The mobile application is a client that consumes the resources of the server. This is necessary to maintain accurate product and pricing details and keep the information on the client side up to date. Even some traditional desktop applications have migrated toward web services. Google maintains their text editor, not as software, on their laptops, but as a service provided through Chrome, their web browser. Changes made to the document are sent to the server for processing. This also supports “cloud” services; making your documents more portable as they are accessible on any device that can access the internet.

Those examples show some of the many benefits of web services. Using XML, it is possible for programs running on many different language platforms to communicate to one another. The architecture of web services is built on the HTTP, and with standardized message protocols like SOAP, REST and JSON, messages between programs can be simple or as complex as needed (What are Web Services? Architecture, Types, Example, 2019). The real benefits of web services rely on the architectures ability to remain platform and language independent (Fowler, 2004).

## Troubleshooting & Consuming Web Services

JAX-WS in Netbeans provides a great built in tester for testing web services. This feature builds a browser page that allows the developer to view SOAP messages, sent and received messages, and page definitions. When working with other web service technologies, the debugger is still a valuable tool. It allows you to access the data that is being sent and the location it is being sent to. More troublesome for the developer is the status of the network. You must ensure that the server is running properly and capable of transmitting the messages.

In our desktop client-server program, web services could be consumed by adding the right web service technologies to the client and server programs. The client, instead of using Sockets, could incorporate JAX-WS. This would allow the client to send SOAP messages to the server. The server, no longer using a Socket for read and write functions, could use web service technologies to support accessing its methods. The client-server Socket connection maintains a constant line of communication between the two. To transition to using web services for the program would be a reconfiguration of the interfaces that the two programs have; a transition from reading and writing objects to requesting and responding to method calls.

## Technologies & Persistence for Web Services

JavaServer Faces: JSF’s are the complete architectural answer to web services. They handle client-side page navigation, all the GUI controls and event handling, and process HTTP requests (Fowler, 2004).

RESTful Web Services: JSF’s use REST for sending requests to the server. In a RESTful web service, methods provided from the server are assigned their own URL. This enables quick access to the service and can provide more flexibility in message responses than the standard SOAP messages.

SOAP: A SOAP message is an XML message sent over HTTP. It’s a standard format, governed by the W3 organization, and allows for the easy transmission of webservices between all types of programs. The use of XML and the standardization of SOAP is what truly makes web services platform and language independent.

WSDL: A Web Service Description Language is a server’s instructions on how to access its methods by using web services. Coordination between developers isn’t necessary if the WSDL is made available to the requesting party. It will contain which methods can be called, what parameters are necessary for successful processing, and what return message the client can expect from the server.

Maintaining Persistence: With JavaServer Faces, a persistence.xml file is created to maintain persistence with the database. This Persistence Unit is typically named *\*ProjectName\**PU and points to a data source. For this project, the data source is the database that stores the products and customers information.

# References

Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standadrd Objet Modeling Language.* Boston, MA: Pearson Education.

*What are Web Services? Architecture, Types, Example*. (2019). Retrieved from Guru99: https://www.guru99.com/web-service-architecture.html