|  |
| --- |
| **OC Pizza Company**  **Information Management System**  Technical Design file  Version 1.0 |
| **Author**  Scott Bolin  *Engineer* |

Table of Contents

1 - Versions 3

2 - Introduction 4

2.1 - Document purpose 4

2.2 - References 4

2.3 - Technology Stack 4

2.4 - Advantages of the Architecture 5

2.5 - Technical Architecture Diagrams and Description 5

3 - Technical Architecture 7

3.1 - General components 7

3.1.1 - Apache Tomcat HTTP Server 7

3.1.2 - JBOSS Application Server 7

3.1.3 - Push Notification Solution for Mobile Platforms 7

3.1.4 - MySQL 8 – Database Server 7

3.2 - Web Site 7

3.3 - Mobile Applications 7

3.3.1 - Android Application 7

3.3.2 - iOS and iPadOS Application 7

4 - Roll-Out Architecture 8

4.1 - Database (MySQL) Server 9

4.2 - Apache Tomcat Server 9

Authentication Framework 10

5 - Software architecture 11

5.1 - General principles 11

5.1.1 - Layers 11

5.2 - Mobile Application 11

5.2.1 - Android app details 11

5.2.2 - iOS app details 11

5.2.3 - Website details 11

6 - Specific points 12

6.1 - Resources 12

6.2 - Development environment 12

6.3 - Packaging / delivery procedure 12

7 - Glossary 13

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Date | Description | Version |
| SB | 03/10/2020 | Document creation | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Document purpose

This document outlines the technical details for the OC Pizza Information Management system.

This document provides the technical details used in developing the OC Pizza Company Information Management System Application.

The elements of document include:

* Technical details
* Roll out instructions
* Architectural Details
* Project Specifics

## References

For further information, please refer also to the following elements:

1. FDF - 1: Functional design for the application
2. OF – 1: Operational File
3. DN – 1: Delivery Note File

## Technology Stack

Proposal utilizes an n-tier architecture consisting of a client tier, a server tier, and a data tier. The client tier contains a PC client (running a standard web browser) and handheld devices. The server tier contains the Web Server and the Notifications server (a framework for managing notifications issued to handheld devices). The data tier consists of MySQL database and storage. See Figure 1 for Backend/Frontend differentiation.

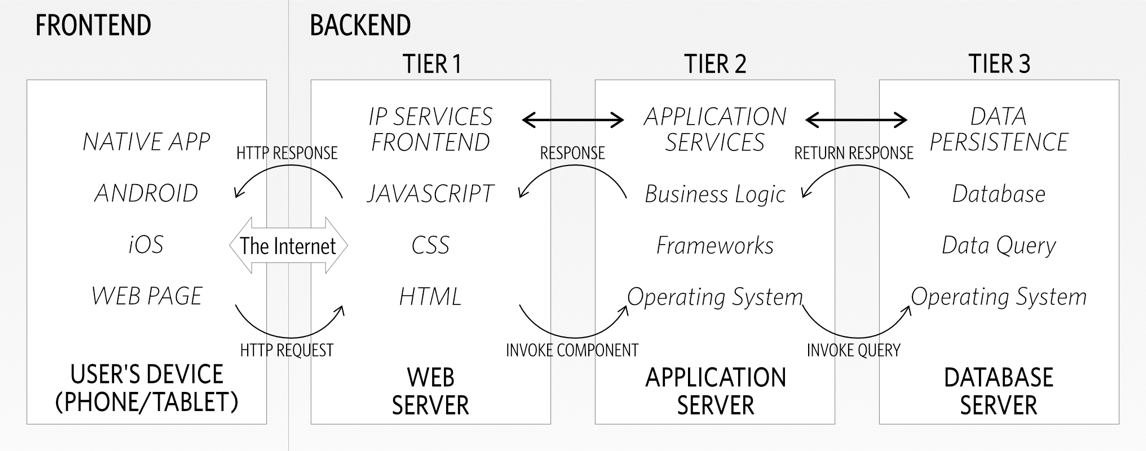


Figure 1 – Backend/Frontend Details

## Advantages of the Architecture

The backend includes a robust distributed computing platform which enables enhanced performance and allows for scalability.

The n-tier architecture allows for the encapsulation of business logic, shielding the client from the complexity of the backend system. Any given tier need not be concerned with the internal functional tasks of any other tier.

The following list is a summary of the advantages using an n-tier architectural design:

* Scalability: Hardware and software can be added to meet retailer requirements for each of the tiers.
* Maintainability: The separation of presentation, business logic, and data makes the software cleaner, more maintainable, and easier to modify.
* Platform independence: The code is written once but can run anywhere that Java can run.
* Cost effectiveness: Open source market-proven technology is utilized, while object-oriented design increases reusability for faster development and deployment.
* Ease of integration: The reuse of business objects and function allows for faster integration to enterprise subsystems. N-tier architecture has become an industry standard.
* High availability: Middleware is designed to run in a clustered environment or on a low-cost blade server.
* Endurance: Multi-tiered physically distributed architecture extends the life of the system.
* Flexibility: The system allocates resources dynamically based on the workload.

## Technical Architecture Diagrams and Description

This section provides a high-level overview of the technical architecture. Figure 2 illustrates the major pieces of the typical three-tiered implementation. Descriptions follow the diagram.

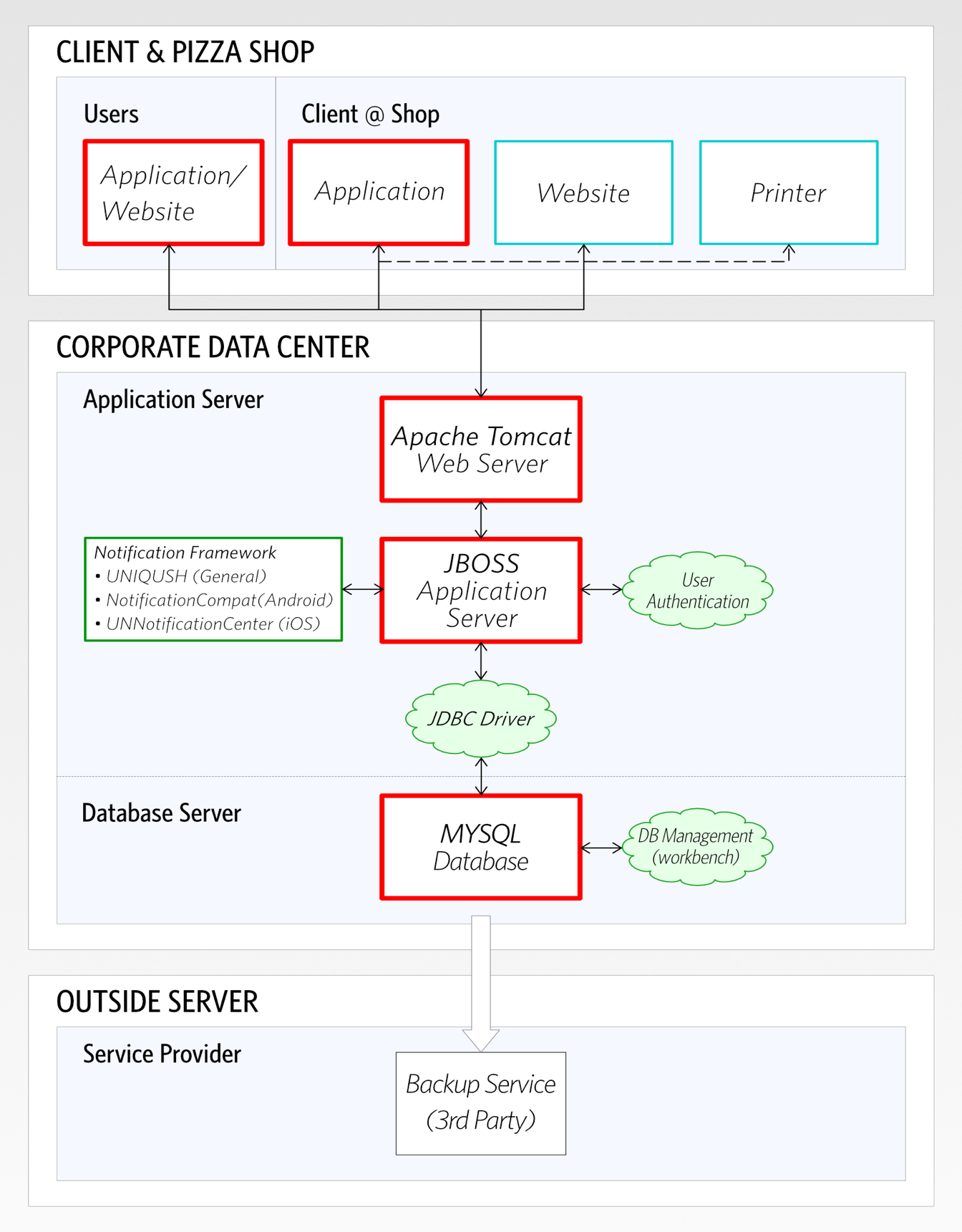


Figure 2 – Architectural Implementation

# Technical Architecture

## General components

### Apache Tomcat HTTP Server

Web server for users to serve HTML documents in a web browser including the OC Pizza main web page, information about OC Pizza company, and order pizza and check order status. In addition, Client management will have their own site served for managing company operations..

### JBOSS Application Server

The application server contains the business logic for running the website and interfacing with the database server to store and retrieve client information, such as employee information, stock status, pizza recipes, customer information such as name, address, contact, past orders, and current order and status is any. Accounting information will be kept separate. This will be run on an independent server (ie, not shared with Database server).

### Push Notification Solution for Mobile Platforms

Typically use Android/iOS frameworks for push notifications, and use Uniqush for internal notifications via web management page.

### MySQL 8 – Database Server

MySQL will be used as the database system. Management of the database will use MySQL workbench. The database will be run on an independent server.

## Web Site

The software stack is as follows:

* **HTML/CSS/Javascript**
* **Apache Tomcat Webserver /JBoss application Server**
* **MySQL Database**

UML Component Diagram

## Mobile Applications

### Android Application

The OC Pizza Company application will run on devices running Android v8 and above. Apps are written using Android Studio IDE using Kotlin. Standard frameworks are used with no external dependencies.

### iOS and iPadOS Application

The OC Pizza Company application will run on devices running iOS/iPadOS version 12 and above. Apps are written XCode IDE using Swift. Standard frameworks are used with no external dependencies (no Swift packages or ‘CocoaPods’).

# Roll-Out Architecture

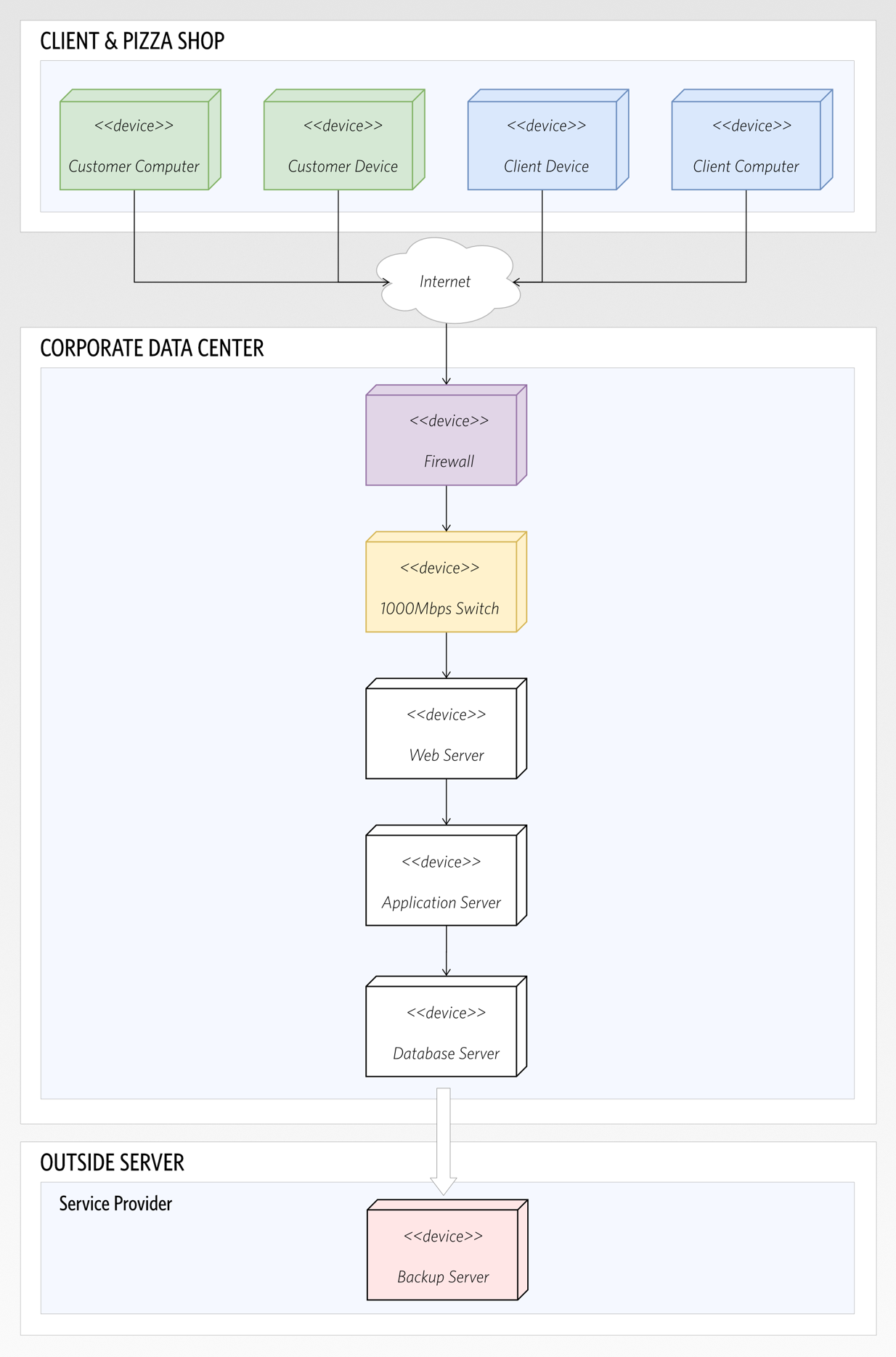


Figure 3 - UML Roll-out diagram

## Database (MySQL) Server

After downloading the MySQL Installer for the first time, a setup wizard guides you through the initial installation of MySQL products. The initial setup is a one-time activity in the overall process. MySQL Installer detects existing MySQL products installed on the host during its initial setup and adds them to the list of products to be managed.

During the initial setup, you are prompted to select the MySQL products to be installed on the host. One alternative is to use a predetermined setup type that matches your setup requirements. By default, both GA and pre-release products are included in the download and installation with the **Developer Default**, **Client only**, and **Full** setup types. Select the **Only install GA products** option to restrict the product set to include GA products only when using these setup types.

Choosing one of the following setup types determines the initial installation only and does not limit your ability to install or update MySQL products for later:

* **Developer Default**: Installs a full suite of applications, examples, and documentation suitable for application development with MySQL:
* ***Server only***: Only install the MySQL server. This setup type installs the general availability (GA) or development release server that you selected when you downloaded MySQL Installer. It uses the default installation and data paths.
* ***Client only***: Only install the most recent MySQL applications and MySQL connectors. This setup type is similar to the Developer Default type, except that it does not include MySQL server, or the client programs typically bundled with the server, such as **MySQL** or **mysqladmin**.
* ***Full***: Install all available MySQL products.
* ***Custom***: The custom setup type enables you to filter and select individual MySQL products from the [MySQL Installer catalog](https://dev.mysql.com/doc/mysql-installation-excerpt/8.0/en/mysql-installer-catalog-dashboard.html#windows-product-catalog).

It is recommended to install the Developer Default upon first run setup.

## Apache Tomcat Server

Download the latest Apache Tomcat server (ver 8.5.58 at time of writing for Windows and Linux systems), and following installation instructions. Installation packages for RHEL (v8.x), Debian (v10.x), UBUNTU (v20.04), SLES (v15), and Alpine (V3.x) distributions are available.

Installation scripts for each installation type are available and should be run prior to installation.

## Authentication Framework

Elytron is a component of JBOSS, an authentication framework which handles all authentication duties for the webserver, application server, and management functions. This framework allows the user to seamlessly login to services using the mobile app or website. All functionality is controlled by the Application Server administrator. See Figure 4 for overview.

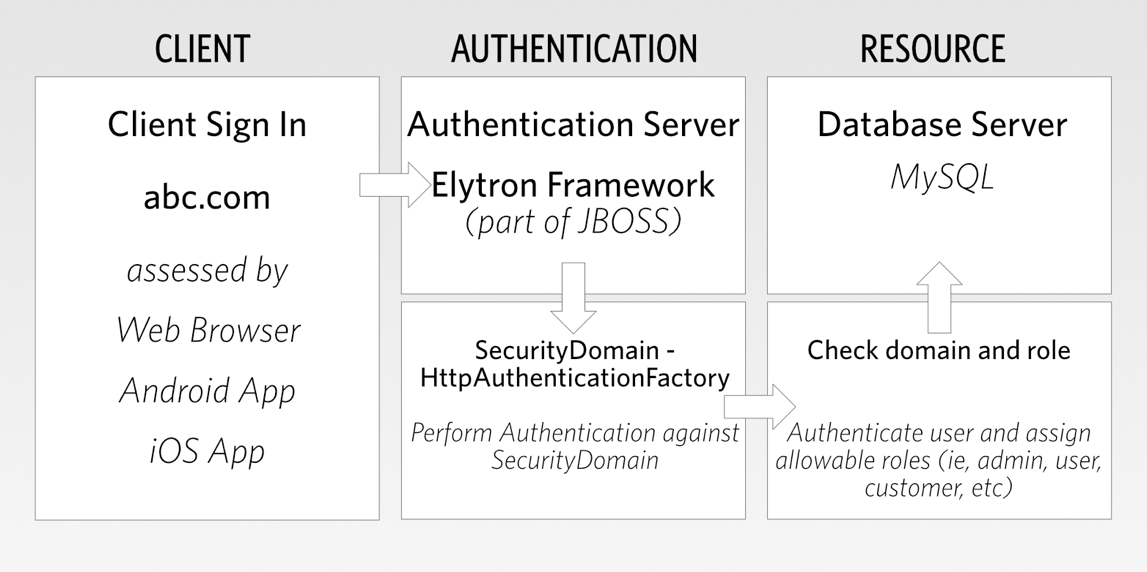


Figure 4 - Authentication Implementation

# Software architecture

## General principles

Project sources and versions are managed by **Git*,*** dependencies and packaging by **Maven/Grunt**

### Layers

Application architecture is as follows:

* A **client** layer is responsible for presentation of information/data to the end user (client), via Mobile Application (Android and iOS), and via web page;
* an **application** (or **business)** layer: responsible for the business logic of the component;
  + a **notification** layer: manage notification events (such as order status, stock status, etc) for both Employees and Customers;
  + an **authentication** layer: Use Elytron framework to handle login authentication and access to specific application and data functions based on user credentials
* a **data** layer: implementation of the business objects model.

## Mobile Application

### Android app details

The OC Pizza Company application will run on devices running Android v8 and above. Apps are written using Android Studio IDE using Kotlin. Standard frameworks are used with no external dependencies.

### iOS app details

The OC Pizza Company application will run on devices running iOS/iPadOS version 12 and above. Apps are written XCode IDE using Swift. Standard frameworks are used with no external dependencies (no Swift packages or ‘CocoaPods’).

### Website details

The software stack is as follows:

* HTML/CSS/Javascript
* Apache Tomcat Webserver /JBoss application Server
* MySQL Database

# Specific points

## Resources

Relevant Links:

Apache Webserver: [Apache Tomcat Webserver](http://tomcat.apache.org/)

JBOSS Application Server: [JBOSS Application Server](https://developers.redhat.com/products/eap/overview)

Elytron Authentication Framework: [Elytron Framework](https://docs.wildfly.org/17/WildFly_Elytron_Security.html#about)

MySQL Database: [MySQL](https://dev.mysql.com/doc/)

Android Studio: [Android Studio](https://developer.android.com/studio)

Xcode: [Xcode](https://developer.apple.com/xcode/)

Website: [Visual Studio Code](https://code.visualstudio.com/)

## Development environment

Android Studio is used for Android development, while Xcode is used for iOS/iPadOS development. Website development uses Visaual Studio Code

## Packaging / delivery procedure

All packages for the project have downloadable binaries for easy installation. All installation files will be downloaded to the appropriate server and installed.

# Glossary

|  |  |
| --- | --- |
|  |  |
|  |  |