|  |
| --- |
| **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 4

2.5 - Technical Architecture Diagrams and Description 5

3 - Technical Architecture 6

3.1 - General components 6

3.1.1 - NGINX HTTP Server 6

3.1.1.1 - Component X 6

3.1.2 - Uniqush – Push Notification Solution for Mobile Platforms 6

3.1.2.1 - Component X 6

3.1.3 - OAuth 2.0 – Authorization Framework 6

3.1.3.1 - Appropriate language (should be built into server, OAuth supports all main languages) 6

3.1.4 - MySQL 8 – Database Server 6

3.1.4.1 - MySQL Workbench 6

3.2 - Web application 6

3.2.1 - Components X 6

3.2.2 - Components Y and Z 6

3.3 - XXX... application 6

4 - Roll-Out Architecture 7

4.1 - Database (MySQL) Server 7

4.2 - NGINX Server 7

4.3 - OAuth Framework 7

5 - Software architecture 8

5.1 - General principles 8

5.1.1 - Layers 8

5.1.2 - Modules 8

5.1.3 - Source structure 8

5.2 - Web Application 9

5.3 - Xxx Application 9

6 - Specific points 10

6.1 - Log Management 10

6.2 - Configuration folders 10

6.2.1 - Web application 10

6.2.1.1 - Data sources 10

6.2.1.2 - Xxx.yyy folder 10

6.2.2 - Xxx Application 10

6.3 - Resources 10

6.4 - Development environment 10

6.5 - Packaging / delivery procedure 10

7 - Glossary 11

# Versions

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Date | Description | Version |
| SB | 14/09/2020 | 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

## 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 Application Server (deployed as a J2EE application) and the Notifications server (a standalone server for managing notifications issued to handheld devices). The data tier consists of MySQL database and an LDAP directory.

## 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.

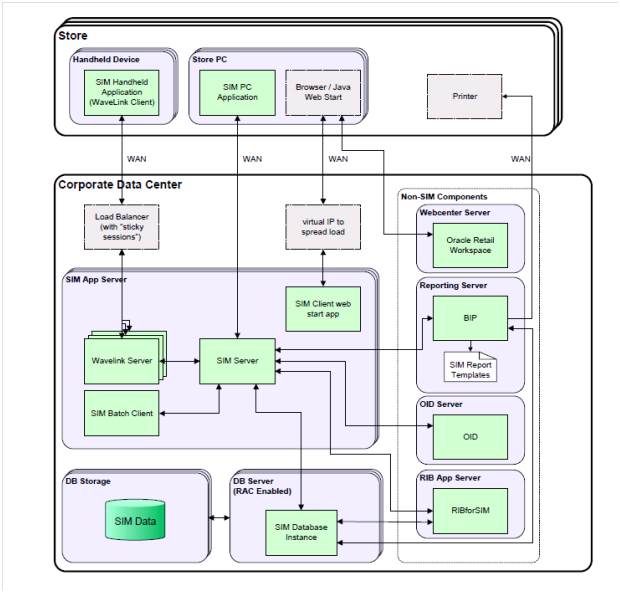
The following list is a summary of the advantages using an n-tier architectural design:

* 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 2-1](https://docs.oracle.com/cd/E12454_01/sim/pdf/140/operations_guide/Output/techarchitecture.htm#CACDAHFC) illustrates the major pieces of the typical three-tiered implementation. Descriptions follow the diagram.

***Figure 2-1 Three-Tiered Implementation***



* **Update Picture**

# Technical Architecture

## General components

### NGINX HTTP Server

#### Component X

Description and role/objective

### Uniqush – Push Notification Solution for Mobile Platforms

#### Component X

### OAuth 2.0 – Authorization Framework

#### Appropriate language (should be built into server, OAuth supports all main languages)

### MySQL 8 – Database Server

#### MySQL Workbench database management UI

## Web application

The software stack is as follows:

* **J2EE** application (JDK version 1.8) / **PHP** (version) / **Python**…
* Application server **JOnAS 5.2.4 / ...**

UML Component Diagram

### Components X

Description and role/objective

### Components Y and Z

## XXX... application

# Roll-Out Architecture

UML Roll-out diagram

## Database (MySQL) Server

Description

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.

## NGINX Server

Download the latest NGINX server (ver 1.19.2 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.

## OAuth Framework

Install framework for particular use case.

# 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 **data** layer: implementation of the business objects model ;
* a **notification** layer: manage notification events (such as order status, stock status, etc) for both Employees and Customers;
* an **authorization** layer: manage user credentials, allowing access to specific application and data functions

### Modules

Maven modules in the case of a multi-module application…

### Source structure

The logic for the structure of the project directories is as follows:

* the source directories are created so as to respect the Maven philosophy (i.e. “convention over configuration”)

root  
 ├─ *pom.xml*  
 ├─ <moduleX>  
 │ ├─ *pom.xml*  
 │ └─ src  
 │ ├─ main  
 │ │ ├─ java  
 │ │ └─ resources  
 │ └─ test  
 │ ├─ java  
 │ └─ resources  
 ├─ <moduleY>  
 │ ├─ *pom.xml*  
 │ └─ src  
 │ ├─ main  
 │ │ ├─ java  
 │ │ └─ resources  
 │ └─ test  
 │ ├─ java  
 │ └─ resources  
 └─ src  
 └─ lib

* ...

## Web Application

…

If needed, UML component diagram to show the various modules and their interdependencies

## Xxx Application

…

# Specific points

## Log Management

…

## Configuration folders

### Web application

...

#### Data sources

...

#### Xxx.yyy folder

...

### Xxx Application

...

## Resources

...

## Development environment

## Packaging / delivery procedure

# Glossary

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