|  |
| --- |
| Codeora |
| Prestige YoYo Production Reporting |
| System Design |

|  |
| --- |
| Thomas Kempton, Hekar Khani, & Samuel Lewis  12/3/2012 |

Table of Contents

[Supporting Libraries 2](#_Toc342234955)

[Entity Framework 2](#_Toc342234956)

[ASP.NET MVC 3 2](#_Toc342234957)

[Ninject 2](#_Toc342234958)

[Highcharts 2](#_Toc342234959)

[jqGrid 2](#_Toc342234960)

[Database Design 3](#_Toc342234961)

[Production Data 3](#_Toc342234962)

[User Data 4](#_Toc342234963)

[Entity Data Model Metadata 4](#_Toc342234964)

[Data Access Layer 5](#_Toc342234965)

[Database Context 5](#_Toc342234966)

[Repository Layer 5](#_Toc342234967)

[Service Layer 5](#_Toc342234968)

[Application Architecture 6](#_Toc342234969)

[Models 6](#_Toc342234970)

[Controllers 6](#_Toc342234971)

[Views 6](#_Toc342234972)

# Supporting Libraries

Our solution is built upon a number of third party libraries that simplify and speed up our implementation process.

## Entity Framework

By using the ADO.NET Entity Framework, we are able to off-load a large portion of our database management and design. The Entity Framework’s code first approach allowed us to create our models based on data as it was read into our application, and store it in a database in the most efficient manner.

## ASP.NET MVC 3

Use of ASP.NET’s MVC 3 framework, we are able to maximize our code reusability, while maintaining a high level of encapsulation within our application controllers.

## Ninject

By using Ninject for inversion of control, we are able to have decoupled classes within our data access layer. This increases our ability to unit test and debug our system.

## Highcharts

The Highcharts JavaScript charting library allows us to provide user friendly and interactive charts to neatly present our data.

## jqGrid

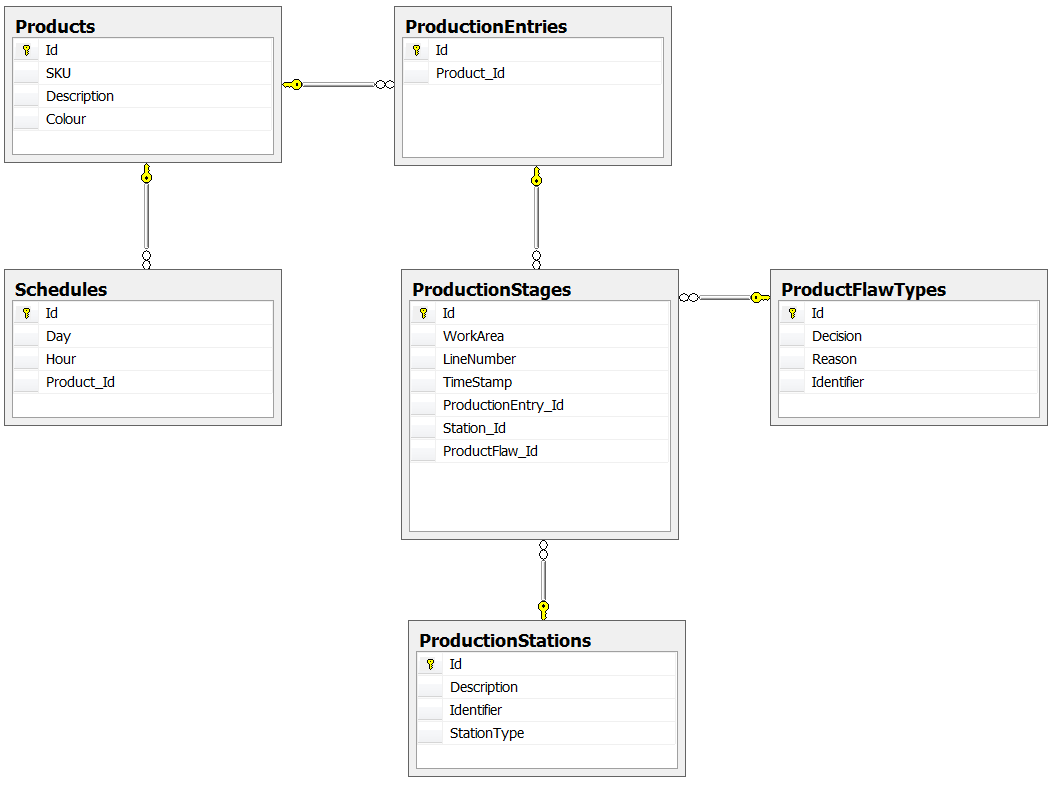
By using the jqGrid JavaScript library, we are able to present potentially long lists of data with support for paging, searching, filtering, and editing, all from a single web page. This is done to speed up use of the application, and to minimize the number of full page loads made to the server.

# Database Design

Our database design currently consists of two separate sets of related tables: production and user data.

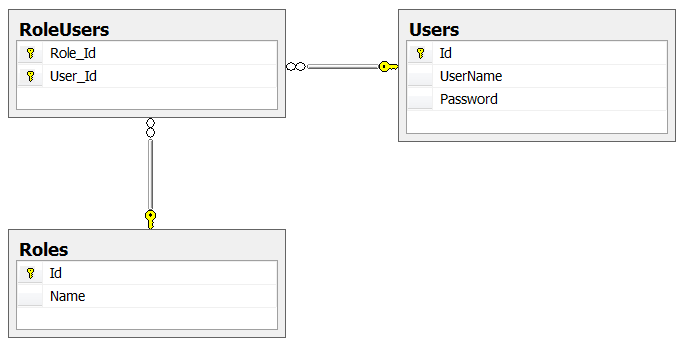
## Production Data

The production data tables are used to store data that is read into the database from the production line/simulator.



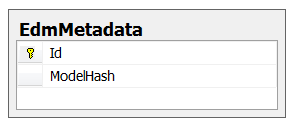
## User Data

The user data tables are used to manage users and their respective roles within the reporting application. Currently, they are not related to the production data tables.



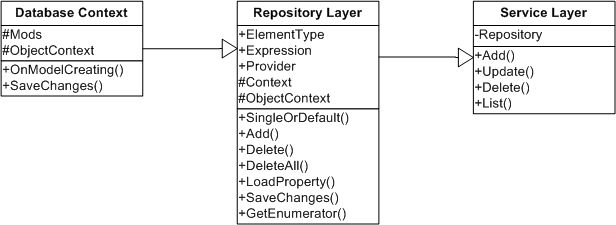
## Entity Data Model Metadata

There is a miscellaneous table that is added by the Entity Framework that is used to track the current version of the database.



# Data Access Layer

The application data access layer consists of a customized database context, a set of repositories, and a set of services.



## Database Context

The database context directly wraps the database and is used for all database interactions. The Prestige Database Context is customized to include features of assigning identifiers to entities as they are added to the database. There is also a module to perform soft deleting, but it is not currently in place.

## Repository Layer

The repository layer wraps a generic database context, and is bound to a specific database model. It provides basic functionality for retrieving and manipulating database entries.

## Service Layer

The service layer consumes one or more repositories and provides basic functionality as well as any necessary business logic – for example, transforming a password into a secure hash.

# Application Architecture

The web application is uses a Model-View-Controller architecture.



## Models

The application model layer is made up of the service classes from the Data Access Layer. These are injected into the controllers by Ninject.

## Controllers

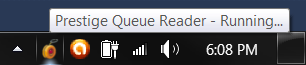
The application controllers perform the tasks of serving data to the user, as well as acting upon the model as specified by the user.

## Views

The view engine transforms the data from the Controllers into a user-friendly format to be presented to the user.

# Production Queue Reader

The production queue is designed to run in the task bar, and has minimal user functionality. It can be stopped, started, and closed.

## Architecture

The queue reader is architected around the reusability of existing code. It is a single application context, and consumes the same data access layer as the web application.