Dosequis.com Tech Specs

# Software requirements

* .Net 4.6.1
* SQL Server 2008
* Visual Studio 2015
* ASP.Net MVC 5 w/StructureMap IOC
* Entity framework 6.1.3

# Visual Studio Project Description

The Visual Studio solution contains the following projects:

**DosEquis.Com**

This is the main website project running Asp.Net MVC 5. It uses StructureMap to manage IOC container and inject dependencies to controller constructors.

The project requires DosEquis.Com.Core, Havas.Social.Aggregator and Havas.Common as dependencies.

The site root is located on staging server at “D:\HTTP\dosequis.com\wwwroot”.

**DosEquis.Com.Core**

The class library project encapsulates all the application/business logic to enable code reuse. The database access is via Entity Framework, and third party API integration code are created as repository and service classes using coding to interface principle.

**DosEquis.Moderate**

This is the web project for “socialadmin.dosequis.com”. It is an old project for social feed and contest entry moderations. Therefore, it contains artifacts of older campaigns as well as some of social feeds that are no longer in use. After the recent site redesign, only the social feeds on dosequis.com homepage are being moderated via this tool.

This site has its own database for enabling and managing user login using standard Asp.Net membership provider. The social feeds and contest entries are pulled in from the main website database.

The site root is located on staging server at “D:\HTTP\socialadmin.dosequis.com”.

**Havas.Social.Aggregator**

This class library project is responsible for aggregating Facebook, Twitter and Instagram based on an user account or hashtag. It makes API calls based on feed configuration xml file and either return or insert into database. Refer to SampleApiConfig.xml at project root for the valid format.

**DosEquis.Soical.Aggregator.Fetcher**

This is a console application that utilizes Havas.Social.Aggregator to fetch and save social feeds into database for moderation. The console is set up in windows task scheduler on dev and staging server to run every morning.

The name of task on staging is “Dosxx\_Social\_Feed”, and is located at “D:\HTTP\dosequis\_socialfeed\_fetcher”. Refer to feedconfig.xml to see the list of social feeds getting pulled.

**Havas.Common**

This is a class library project that contains generic, common utility classes and static functions to promote code reuse.

# Front-end Setup and Configuration

Dosequis.com is Fully responsive site targeting modern browsers ie11+, Chrome/FF/Safari (last 2 versions).

This site was built using [Webpack](https://webpack.js.org/) to compile modular JS using ES6 and CommonJS syntaxes. Babel is used for polyfills and to allow writing JavaScript using ES6. CSS is bundled using SASS and Autoprefixer.

**Requirements**: node v4.4.5+

**Installation**: - all npm commands may require sudo

Run ‘npm install’

**Production Buld**

Run ‘npm run prod’

**QA Buld**

Run ‘npm run build’

**Local Dev:**

Run ‘npm start’

Since this is a .NET project IIS must be installed locally to get some of the benefits of Webpack Dev Server.

* Set up IIS to point to port 9090
* Run ‘npm install’
* Run ‘npm start’

This will run a proxy server on port 8089 which will automatically inject CSS changes to the page without reload and will automatically refresh the page on any JS changes.

**webpack.config(s)**

There are 3 configuration files for the 3 different environments plus a webpack.parts.js which contains some common variables and the JS/CSS entry points

**Folder Structure:**  
/dev - all front end JS and CSS source files are in this folder  
/\_build - all compiled js and css files are created in this folder. The build folder is completely removed as part of the build processes to ensure all files are up most recent

/Views/Shared/ - all layout template files

/Views/\* - all page template files are in a folder matching the controller name

### Page Specific Code Using External API or Services

**Beer Finder**

VIP services: this is a vendor service arranged through heineken’s IT. They have up to date information on what beer are in stock in locations across the country. We created a back end service which hits their API and sends back JSON data to the front end for the beer finder page.

Google Maps/Google Places:

We are using the Google Maps API to render the locations returned from VIP API calls. We also are using google places to find user's location and to allow the autocomplete search.

API calls are made to VIP for each map tile visible on the page and is updated whenever the map is moved or resized. Map clustering being used to group together any areas that have 4 or more locations.

Filters are in place to search by location type (store/ bar - restaurants) and by product type.

Chrome now enforces HTTPS to get the location from the browser so this page has a redirect in place to force the user to HTTPS.

**Most Interesting Man**

This page uses Google Maps API to render locations on a map from a JSON file. This scroll of the page triggers events depending on what item is currently in view and moves the map to that location. It also allows clicking on the map to move the page to the item featured in that location.

**Adobe Animate CC**

All global touts and most of the animation on the history page are created using Adobe Animate CC

Instructions for updating the various animations created with Adobe Animate.

* **Files**: /dev/animate/
* Instructions:
  + Update the FLA and publish. There are **two** lines of code that need to be changed in the auto-generated .js file
    - 1. The manifest property on the lib.properties object. Copy the manifest object from the old file and replace in the new file. This will make sure your code points to the correct image files on the server.
    - 2. The code at the bottom of the file. Delete the last 2 lines of the generated JavaScript. Copy over the last line from the file on the Dos project. This will look something like this: })(window.beerFinderToutLib = window.beerFinderToutLib||{}, window.beerFinderToutImages = window.beerFinderToutImages||{}, createjs = createjs||{}, window.beerFinderToutSs = window.beerFinderToutSs||{});
  + In the Dos project, the .js files for the animations are located: **dev > scripts > components > touts** and **full-page**
  + The filenames will either have **\_export** or **\_tout** on the end

# Staging and Production Environment

The host provider is Striata who performs system administration tasks such as patching the server, setting up backups, IIS, DNS etc…

There is one staging server and 3 load balanced live server. The database is hosted on a separate server.

### **Client staging**

* [http://staging](http://staging/).dosequis.com
* FTP and RDP access
  + Host: staging.usa2.heineken.com
  + User: euro
  + Pass: nacREb6u\*e

### SQL server access:

* + host: db.usa2.heineken.com
  + user: eurosql
  + pass: xAz73a6rUf

### **Live servers**

* 3 load balanced servers - backdoor URL to each instance
  + [http://lb1](http://lb1/).dosequis.com
  + [http://lb2](http://lb2/).dosequis.com
  + [http://lb3](http://lb3/).dosequis.com

Live URL http://www.dosequis.com

# Deployment Instructions

#### **File/Code Deployment**

1. RDP into client staging (staging.usa2.heineken.com)
2. Go to D:/Http/dosquis.com/wwwroot, move web.config out of web root, otherwise staging web.config will get deployed to live servers
3. Get out of web root and back to D:/Http/, run rsync\_dosequis.bat
4. Set "Delete files from production which don't exist on staging" to "N" to prevent deleting web.config from live servers
5. Do a "dry" run if needed, at the end of script it will sync the files between staging and live
6. Copy web.config back to wwwroot to activate the staging site
7. Jump on to backdoor live server URLs to verify all the files have been synced between staging and 3 live servers

#### 

#### 

#### **Database/CMS Deployment**

The database changes can be deployed via SQL scripts, or alternatively use Redgate’s SQL and SQL data compare tool.

# API/SOCIAL APPS

Facebook

https://developers.facebook.com/apps/209002842602792/dashboard/

Instagram

https://www.instagram.com/developer/clients/a47ade608e3042f1927f3bc447208046/edit/

Google API

MIM map

<https://console.developers.google.com/apis/dashboard?project=dos-equis---mim-1475525534069&authuser=2&duration=P30D>

Beer Finder  
<https://console.developers.google.com/apis/dashboard?project=dos-equis-beer-finder&authuser=2&duration=P30D>

Twitter:

Token does not need to be tied to specific app. If changes need to be made create new app to get token.