
Food Ordering System for the GMIT Catering Company

Ronan Connolly

Vladislav Marisevs

B.Sc.(Hons) in Software Development

APRIL 21, 2016

Final Year Project

Advised by: Dr John Healy

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)



Contents

1	Introduction	7
1.1	Chapter Summaries	8
1.2	GitHub Links	9
2	Context	11
2.1	System Administration and Management Web Application . .	11
2.2	Mobile App	11
2.2.1	Cross Platform Frameworks	12
2.2.2	Tooling	12
2.3	Push Server	13
3	Methodology	15
3.1	Direction	15
3.1.1	Interface	15
3.1.2	Bleeding Edge Technologies	15
3.2	Agility	15
3.2.1	Agile	15
3.2.2	Meetings	15
3.2.3	Team Work	15
3.3	Testing	15
3.3.1	Test Server	15
3.4	Source Control	15
3.4.1	GitHub	15
3.5	Choice	15
3.5.1	PHP	16
3.5.2	SQL	16
3.5.3	JavaScript	16
3.5.4	JSON	16

4	Technology Review	17
4.1	System Administration and Management Web Application . .	17
4.1.1	php	17
4.1.2	Zend Framework 2	17
4.1.3	MySQL	18
4.1.4	Composer	18
4.1.5	WAMP Server	19
4.1.6	Server Grove Hosting	19
4.2	GitHub	20
4.3	Mobile App	22
4.3.1	Ionic Framework	26
4.3.2	Tools	27
4.3.3	Alternatives	27
4.4	Push Server	27
4.4.1	CouchDB/Cloudant	27
4.4.2	NodeJS	27
4.4.3	ExpressJS	27
4.5	JSON REST interface architecture	28
4.5.1	HTTP Requests	28
4.5.2	Custom API	28
5	System Design	29
5.1	Database	29
5.1.1	Purpose	29
5.1.2	Procedures and Functions	29
5.1.3	Tables	32
5.2	System Administration and Management Web Application . .	36
5.2.1	Admin Authentication	37
5.2.2	Voucher system	37
5.2.3	Ionic routes & functions	38
5.2.4	PDF make	38
5.3	Mobile App	38
6	System Evaluation	39
6.1	Web App	39
6.2	Mobile App	39
7	Conclusion	40

About this project

Abstract This project sets out to create a food ordering system for a local company. The systems primary components are a mobile application that the user interacts with and a web application that the staff interact with.

The need for such a system stems from two problems, firstly the issue of rush hour times during business hours where there is a vast number of customers to service, and secondly to bring more presence and promotion to the business, as they are finding it hard to reach out to their current customer base and would be customers.

We aim to solve the first problem by having a system in which customers can pre-order sandwiches and other products via a mobile application. Users will be able to top up their account, order products, pick a collection time, and view their balance, products and past order history.

The second problem will be solved by implementing push notifications into the application so that the company can let customers know about menus, events and various other updates. Another way to increase promotion and presence is by having various information about the company on the application; for instance: opening times, contact details and general information.

All of the information on the web application can be updated; this includes the menus, opening times, user and staff details, and much more. This information is reflected in the web application. Interactions from within the mobile application including: topping up, logging in, registration and ordering go through the web application.

We plan to create a cohesive, thoughtfully designed, robust system that solves these two problems.

Authors This project was created by two fourth year software development students: Ronan Connolly & Vladislav Marisevs, as part of our Bachelors of Science honours degree in Software Development.

Ronan was in charge of creating all aspects of the user facing mobile app. Vladislav was in charge of creating all aspects of the staff facing web app.

We spent most of our shared time coming up with the overall architecture we would implement, and an interface to be used between mobile and web app for transfer of data.

Acknowledgements We would like to acknowledge and thank our supervisor Dr John Healy for all the time and effort he has put into helping us throughout this project, he gave us a good structure and set milestones for us in order to keep on top of things.

We'd also like to thank the GMIT Catering Company staff for the time spent meeting with us in order to continuously improve and adapt the project.

Chapter 1

Introduction

We set out to create a food ordering system for the GMIT Catering Company (known henceforth as the company). The basic structure is a mobile application (henceforth known as mobile app) for the Android and iOS systems that the user interacts with, and a server (henceforth known as web app) that the staff can log into in order to view transactions, user details and to update the mobile app.

The reason such a system is needed is that queues during peak times tend to be enormous and currently it's hard to service all the customers.

Another reason is to encourage customers to get into a habit of repeat ordering, if it is an easy process then it should increase purchases.

Lastly, the company wants to increase presence and promotion in the college, in order to achieve this end we have implemented push notifications where staff can send a notification to all users. On top of this the mobile application itself serves as a promotional device, containing details of various aspects of the company.

In order to develop this system we required to connect different platforms together and let them communicate. We were using Ionic Framework for creating cross mobile application that would talk to Zend Framework 2 which will act as administration website and API (Application Programming Interface) for controlling data transfer between MySQL database and mobile program.

The components contained within the mobile app include pages for login, registration, about the company and user details. There is also a way to top up and order sandwiches. A huge emphasis is put on design for this project, using the company's colour theme and creating a nice icon. This mobile app was created using the Ionic Framework which is programmed primarily using

the AngularJS framework. It's a cross platform mobile application that will authenticate users using their credentials. This option will allow us to create an account wallet, identify a person and their order history. The mobile app design uses native phone features and user interface components to let user operate without any special training.

The components contained within the web app include many pages such as the login system, orders, stock, user details, vouchers(for adding credit to your account), settings(collection and opening times) and accounts(staff) pages. This web app is a administration website which allows user to configure and manage the whole system. Users can change opening, closing and food collection times. In order to access these settings users should authenticate him self and then will be able to nominate new administration members. This website also allows to view list of orders, customers and track their history.

This web app was created in PHP using Zend Framework 2. Most of the information on the web app is reflected on the mobile app.

The two applications talk to each other via JSON over HTTP Get and Post requests.

We set out to create a well thought out, carefully designed, robust food ordering system using modern technologies. This project could be extended in the future to be used

We used an agile structure where we had certain components we needed complete by specific dates. We had various meetings each month with our supervisor and several members of the company.

1.1 Chapter Summaries

Here is a summary of all the chapters in this project report.

Context

Here we talk about how the project came about, what the initial ideas, goals and objectives were. We'll also talk about the main components of our project, how we chose them, the alternatives we tried out and a basic overview of the usage of our food ordering system.

Methodology

An insight into how we began the project, the research, our technology and design choices, our thought process and how we went about allocating tasks and organising meetings.

We'll also talk about the objectives we set out to accomplish and how we went about completing them.

Technology Review

Here we talk about all the technologies that we have mentioned in this report and any others that we may have used in completing the project.

System Design

An overview of the project architecture, including lots of diagrams and screen shots.

System Evaluation

An evaluation of our various project components, including how we tested our system for robustness and performance. We talk about the outcomes that were achieved in relation to what are goals were, how far we strayed from our goals and some issues we came up against. Any limitations or opportunities we encountered in our approach and in the technologies we chose.

Conclusion

A broad overview of our development work-flow, from our thought processes, meetings, technology choices to our issues, problems and perceptions of the project as a whole. We'll touch on our overall experience and what we would have done differently.

1.2 GitHub Links

The web and mobile app repositories are private so you must ask to be added as a collaborator in order to view them. Each of the headings are clickable, they contain links to each GitHub repository.

Gist ReadMe

This contains basic instructions for using each component of the project.

Web App

The food ordering web app.

Mobile App

The Ionic mobile application.

Test Server

The test server that we used initially to test the mobile app. This received requests and sent back mock data that imitated the real server.

Push Web App

This web application is a MEAN stack application. It is hosted on Heroku and is used to save user details. Administrators can log in and send push notification messages to registered users.

Project Report

This project report is located here.

Chapter 2

Context

- Our project consists of creating a food ordering system for the GMIT Catering Company.
- Our basic objectives were to create a mobile and web application to deal with the above item.

Below we will:

- explain the various components of our project.

2.1 System Administration and Management Web Application

I decided to do this because of that, etc PHP is great but maybe a Java web server would have being better, easier to modify, I already know Java. SQL is hard to change over time, maybe implementing a NoSQL database would have been good...

2.2 Mobile App

The mobile application is created using bleeding edge technologies such as the Ionic framework, which utilizes the AngularMVW framework, which in turn is programmed using JavaScript. HTML and CSS were also heavily utilized.

Initially I had no idea about JavaScript, web development or cross platform development. I spent much time studying JavaScript, Angular, Ionic, and the MEAN Stack (MongoDB, ExpressJS, AngularJS, and NodeJS). I

then spent a lot of time trying out various cross platform development frameworks.

I will speak more in detail about these technologies in the technology review chapter.

2.2.1 Cross Platform Frameworks

I first tried out Cordova, which I found had the capabilities to do pretty much anything, but the community and framework is very sparse, it's hard to get anything up and running, and the stylings are horrid. [?]

Next I tried out JQueryMobile which had better styling but again I came across many issues. [?]

Finally I came across the Ionic Framework which uses Cordova underneath. This framework has a huge community of developers, it has the support of Google, Microsoft and many other large companies. They closely work with the Angular team at Google and the TypeScript team at Microsoft. They have a 24/7 group chat system set up with various sub rooms. They have amazing documentation, regular blog posts, quick response to questions on the blog, forums and chat. With Ionic you get:

- All the capabilities of Cordova, which allows you to access the Mobiles native APIs easily
- A slick native UI experience. The app changes design depending on the platform it's running on
- Extensive tooling. Starting an app, templates, app store image creation, logo creator, etc
- Rapid development cycle, there are constant updates (which can cause issues, but is usually great)

[?]

2.2.2 Tooling

Once I decided on using the Ionic framework I spent my time completing various JavaScript, Angular and Ionic tutorials. This was a steep learning curve as there is so much tooling for all the JavaScript frameworks. I installed NodeJS in order to use NPM (Node Package Manager) in order install Ionic.

Then Ionic came with it's own tools for various tasks, including SASS for programmatic CSS, Bower (Like NPM or Maven) for adding in new components, Grunt for running tasks (like Ant) and some others like Gulp (similar again to NPM).

Each of these tools takes time to learn. I read documentation and completed at least one tutorial for each.

2.3 Push Server

In order to facilitate push notifications I needed to get their device token and save it in a database. I created a controller in the mobile application, once the app starts it gets the device token along with some other information and sends it to the push server.

The push server is always listening for incoming requests, once it receives one it evaluates it, adds a timestamp and saves the user object (JSON) to a CouchDB server (Using IBM's Cloudant Web Server).

The push server is a MEAN Stack, Yeoman scaffolded project that uses NodeJS as the environment, Express as the Web Framework and Angular as the MVC (Model View Controller) framework in order to build the web application.

The reasons we chose to have the push notification server separate is that:

- We did not realise we needed to save the device tokens initially and to implement this new table into the SQL schema is a big effort
- We wanted to try out a MEAN Stack application
- We felt there was no big disadvantage on having the push server separate
- The mobile app is so tightly integrated with the push server (everything is written in JavaScript), that it made creating the web application extremely simple
- The GMIT Catering company may assign the task of push notifications to somebody that they do not wish to have access to the food ordering system, such as a social media expert.

Users can:

- Login/Logout via username and password

- View how many devices are registered for push notifications
- View previously sent push notifications
- Send new push notifications, and see if they were sent successfully.

Chapter 3

Methodology

3.1 Direction

3.1.1 Interface

3.1.2 Bleeding Edge Technologies

3.2 Agility

3.2.1 Agile

3.2.2 Meetings

3.2.3 Team Work

3.3 Testing

3.3.1 Test Server

3.4 Source Control

3.4.1 GitHub

3.5 Choice

We already mentioned in the Context chapter about why we chose Ionic, Zend and Mean stack framework/architectures. Here we will discuss why we chose PHP, SQL, JavaScript and JSON as our core technologies.

3.5.1 PHP

3.5.2 SQL

3.5.3 JavaScript

3.5.4 JSON

Chapter 4

Technology Review

Here we will talk about all the technologies we used.

4.1 System Administration and Management Web Application

After gathering all system requirements and we have done a research and based on them I decided to use PHP based framework because there is a lot of hosting companies that support this server scripting language and it has very cheap pricing.

4.1.1 php

PHP is in the market since 1995 and is one of the widely used server scripting language. It is quite powerful tool for making dynamic web pages and it is free. There are many popular websites that are written on *php* such as Facebook, Wikipedia, Flickr, Mailchimp, Wordpress.com. PHP code can be embedded into html page. This type of structure allow to use it in various web template systems, content management systems or web frameworks. From beginning *php* wasn't object oriented language, but in the later versions they redesigned it.

4.1.2 Zend Framework 2

Prototype was developed using *Zend Framework 2*. This is an open source framework for developing Web applications and services. It was written on *php* and it is loosely coupled architecture, which allow developer to

code each component independently and designed with Model View Controller structure. **Zend Framework 2** uses 100% object-oriented code and utilises most of the new features of **PHP 5.3**, namely namespaces, late static binding, lambda functions and closures. [1]

Barry and Elhakeem [2] argues that Zend Framework enables simple, rapid and agile web application development process, and it also offers AJAX support to convert XML data into JSON format and integrates the most widely used APIs and Web Services of third-party companies such as Google, Microsoft, Amazon, Flickr and Yahoo. ZF provides many options for validation such as Dojo tools it also allow to filter all inputs.



Based on Hyun Jung La and Soo Dong Kim publication we can say that this project has a Balanced Model View Controller Architecture [3]. The **Zend Framework 2** contains applications business logic or Model. And it is also responsible for validating user inputs or Server Side Controller and Web application Views. While **Ionic** is responsible for Client Side View and Controller (on the smartphone). In this case Mobile application uses **zf2** Controllers to connect to **MySQL** database.

4.1.3 MySQL

To store information we were using **MySQL** database management system. It's reliability is proven through the years. **MySQL** offers complete ACID (atomic, consistent, isolated, durable) transaction support, unlimited row-level locking, distributed capability, and multi-version transaction support where readers never block writers and vice-versa. Easy manageable platform that allow a DBA to manage, troubleshoot, and control the operation of many **MySQL** servers from a single workstation. [4]



4.1.4 Composer

For such large distributed projects like this more often used some sort of dependency managers, that allow to force the development cycle and reuse any of already written packages. One of the most popular dependency manager for **php** is **composer**. After installing it we can start using it, simply by creating *composer.json* file which describes the dependencies for your project and may contain other metadata as well. [5]



composer.json example:

```
{
  "require": {
    "php": ">=5.5",
    "zendframework/zendframework": "~2.5",
    "zendframework/zftool": "dev-master",
  }
}
```

As you can see, *require* takes a key value pairs where key maps to package names and value is version of it. We also can configure the minimal supported release or nominal version. When *composer.json* is completed, we can run command that will download chosen packages and configure them in our project. To do that run this command in console:

```
$ php composer.phar update
```

4.1.5 WAMP Server

4.1.6 Server Grove Hosting

There is not so many hosting companies that supports Model View Controller structured *php* websites, but the good old days when we were querying **.php* files are gone. This type of structure hides the extension and it will confuse the attacker on what sort of platform it was written.

One of most popular hosting companies in Ireland is Blacknight Solutions. For one of my portfolio websites I was using their services. At that time I was not bothered with developing my own website from the scratch and I have used *Drupal* Content Management System. It is quite easy to install and setup CMS for lite blog type of website, but when it comes to connecting 2 different platforms it is no longer useful. When I developed my portfolio website on *zf2* and tried to host it on their server I had a lot of issues with their policy. Customers don't have access to main *php.ini* file which can change write properties that has to be on to use MVC structured websites.

To solve this problems I have found *Server Grove* hosting company that fully supports *Zend Framework 2*, *Git Client* and *composer*. It took me a while to move my domain name from BlackNight Solutions, but once I have it done all technical support on Server Grove is brilliant.

4.2 GitHub

We used GitHub [6] throughout our project in order to utilize source control and keeping track of versions. We found that it was very useful in that we could have separate repositories for the different parts of the project. When we did work on the same repository we each had our own branch to which we would merge intermittently into the *master* branch. We also used the *issue tracker* to raise any issues/bugs we noticed on each others projects, or to mention an enhancement that could be incorporated.

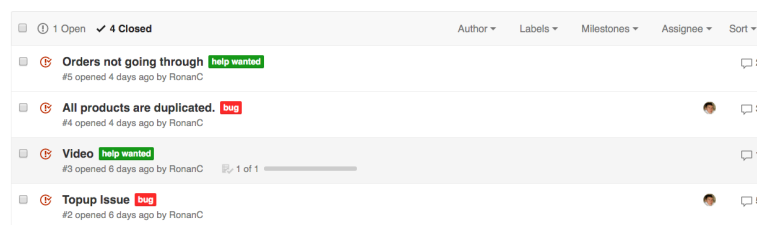


Figure 4.1: Issue Tracker

We also created an organisation, which acts as an entity. We forked all our repositories there and used it as a central hub for the overall project [7].

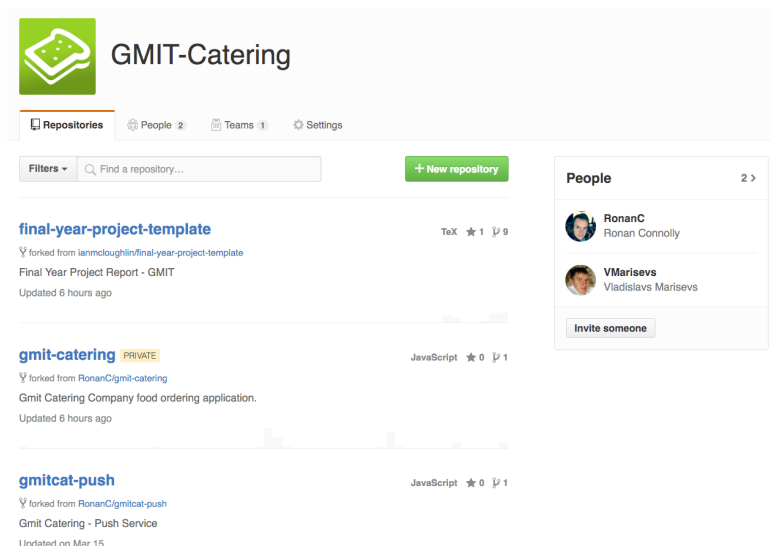


Figure 4.2: GMIT Catering GitHub Org

4.3 Mobile App

The technology stack I chose to use for this project is quite extensive. Ionic is heavily based in the JavaScript community which is known for its enormous amount of libraries and dependencies. It is quite a steep learning curve to get to grips with the basics, and it takes quite a while to get comfortable in the workflow. I came across a tweet that sums up this experience:



Figure 4.3: JS Developer Learning Curve

I done extensive research on different cross platform frameworks. First I started with JQueryMobile and PhoneGap which had the capabilities but the community and tooling were quite scarce.

Xamarin was another option, It did not appeal to me as much, during my research I found many people had lots of issues with it and you still have to write some specific code for each platform.

A lecturer of mine mentioned the Ionic framework to me and once I looked into it, it was the obvious choice. Ionic has a huge community, extensive tooling for the majority of tasks needed and it is rapidly becoming the number one option for cross platform development.

Once I had decided to take the Ionic route it meant that I was delving into the world of web development, as Ionic uses AngularJS, which is used with JavaScript, HTML, CSS and all the web development tools and work-flows.

HTML/CSS

I found that learning HTML [8] and CSS [9] was fine, I just needed to know enough to use Angular, which is very basic. I had done some HTML and CSS before so I just needed to brush up on my skills. I took the Web Development [10] course on Code School which helped me quickly get up to speed.



HTML is great because it is a simple structure that is easy to understand. It is just a subset of XML for websites.

CSS makes it really easy to change various parts of the HTML from one place. I used SASS (Sassy CSS), this lets you use CSS in a programmatic way (using variables).

Here is a piece of SASS for the Ionic colours:

```
$light:                #fff !default;
$stable:               #f8f8f8 !default;
$positive:             #5475aa !default;
$calm:                #11c1f3 !default;
$balanced:            #7baa1e !default;
$energized:           #ffc900 !default;
$assertive:           #840018 !default;
$royal:               #886aea !default;
$dark:               #444 !default;

// The path for our ionicons font files
$ionicons-font-path: "../lib/ionic/fonts" !default;

// Include all of Ionic
@import "www/lib/ionic/scss/ionic";

// custom styles
@import "scss/styles";
```

You can also see that I am importing other style sheets.

JavaScript

Once I had gotten myself into the basic mindset of HTML and CSS I decided to learn JavaScript. I completed three JavaScript courses [11] on Code School. This gave me a great understanding of JavaScript [12].

I found that JavaScript's event based nature took awhile to get the hang of but once understood it was very useful. Promises are a very important concept to understand as JS is asynchronous and tasks may not execute in order, especially blocking tasks. This is useful as you don't need to code in concurrency, it is an inherent feature of the language.

I wrote a literature review on using JavaScript for full stack development [13], this touches on promises, the event based nature, and why JavaScript is so popular.



AngularJS

I completed two AngularJS [14] courses on Code School. Then for each course I completed I done the 2/3 hour screen cast that accompanied it.

The main problem I found with JavaScript is a lack of standards and structure. This is why I believe there are so many JS frameworks out there.

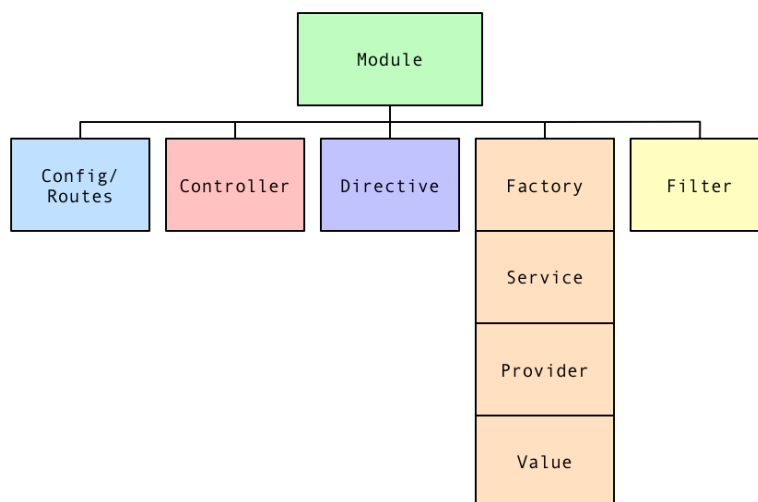


Figure 4.4: Angular Object Diagram

A few times a year there a a new JS framework that all the web developers think is going to be the **one**. AngularJs

has risen and is now one of the top JS frameworks, partly I'm sure to do with the fact that Google is the project owner. Competition includes Facebook's React [15] and EmberJS [16].

I completed two Angular courses on Code School. Then for each course I completed I done the 2/3 hour screen cast that accompanied it.

These helped me gain a deep understanding of controllers, services, routing and views. Angular is based around the Model View Controller (MVC) model, where the design, data and logic are all separated. Angular is a Single Page Application (SPA). Only one page ever exists, the *index.html*, then each *view* is loaded into a section of this page. Each *view* has a *controller* attached to it. The Modal contains the data which is loaded into a template that you code, this creates the view.

Two way binding is used so that as you enter information is can be seen straight away.

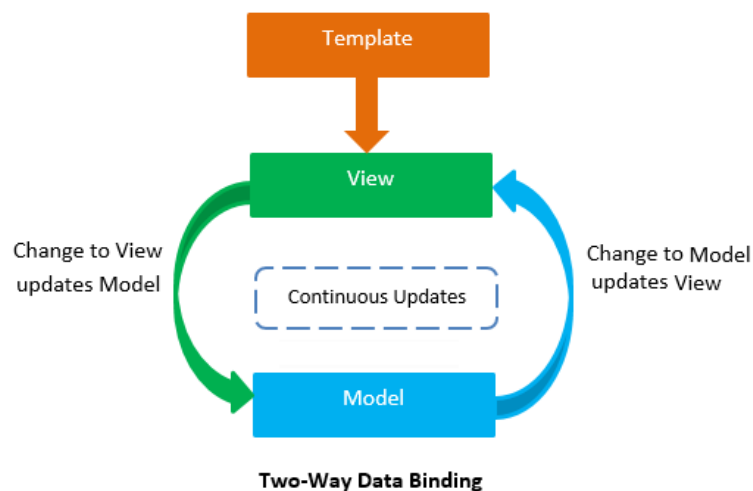


Figure 4.5: Two Way Data Binding

4.3.1 Ionic Framework

Once I understood AngularJS I completed some Ionic tutorials on EggHead.io and Thinkster.io.

The Ionic community is really great, the forum is very engaging, questions are answered very quickly and they post to their blog regularly. The founders Max Lynch, Adam Bradley and Ben Sperry talk about various updates every few months on YouTube, they call this *The Ionic Show* [17].



To Start an Ionic application you run these commands in bash:

```
$ ionic start myApp tabs
$ cd myApp
$ ionic platform add ios
$ ionic build ios
$ ionic emulate ios
```

Everything is done through command line.

Here is a simple controller:

```
(function() {
    'use strict';

    angular
        .module('canteen.controllers')
        .controller('AboutCtrl', about);

    about
        .$inject = ['$scope', 'Orders'];

    function about($scope, Orders) {
        $scope.opening = Orders.opening;
        $scope.closing = Orders.closing;
        $scope.times = Orders.collTimes;
    }
})();
```

One issue I came across is that there is no definitive way to lay out structures. This is a common issue in JavaScript, since the standards are so loose (which benefits it in order ways).

The above layout follows John Papas AngularJS style guide [18]. John Papa works with the Microsoft team who are implementing strong typing into Angular2 (through the use of MS TypeScript). This gives me the impression that he has a good grasp on best design practices.

As you can see in the above example, I have broken up the code into its constituent components, rather than the usual nested JavaScript hierarchy. I think it is pleasing to the eye and easy to understand.

4.3.2 Tools

Yeoman

Grunt

Jasmine

Karma

Bower

NPM

BASH

4.3.3 Alternatives

Cordova

PhoneGap

JQueryMobile

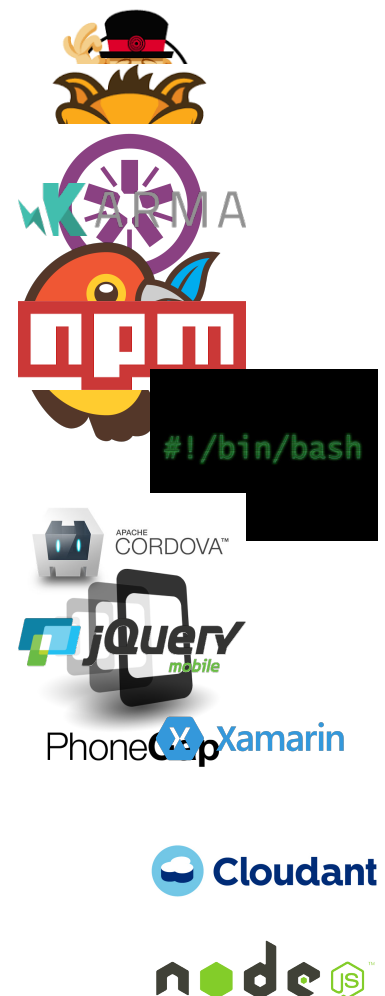
Xamarin

4.4 Push Server

4.4.1 CouchDB/Cloudant

4.4.2 NodeJS

4.4.3 ExpressJS



4.5 JSON REST interface architecture

express

4.5.1 HTTP Requests

4.5.2 Custom API

[19]

Chapter 5

System Design

5.1 Database

5.1.1 Purpose

This section describes how the database that will support the system with details of the logical and physical definitions. It also provides the functional and non-functional usage of the tables, considerations and requirements.

The database design for the system is composed of definitions for db objects derived by mapping entities to tables, attributes to columns, unique identifiers to unique keys and relationships to foreign keys.

Further in this chapter I will describe the integration aspects of the Database with the Web Application and to understand them I want to explain some word definitions that will be used.

- Customer is a physical person who uses smartphone application.
- User is a physical person who operates with management website.
- Food is entire product of this business.
- Voucher is a printed card with unique code that allow customers to top up their account balance.

5.1.2 Procedures and Functions

One of good reasons to use stored procedures is the added layer of security that can be placed on the database from the calling application. The risk of a attacker using the account to run a stored procedure that has been written by you is far safer than having the user account have full insert, update

and delete authority on the tables directly. Another advantage to use stored procedures is the data functionality that is separated from the application making it easier to manage, document and maintain. They also improves the performance for example to make a payment I am doing just one call and DBMS can process it, which is more efficient than do complete processing on the php side.

- **Verify Customer**

This procedure takes two parameters login name and password, then parses it and returns the result. This procedure is used to connect Ionic application with DBMS and verify customer's credentials. If the password and login matches database returns complete details about current customer. This information can be displayed in the mobile application.

- **Block Customer**

If system administrator sees that some user tried to enter invalid voucher id too many times, he can block this person. To keep a track of all invalid customer inputs I decided to add a basic counter, so that administrator can keep an eye on the malicious users that tries to insert voucher code permutations.

- **Change customer's password**

One of the common procedures when user is allowed to change their password if they are authorized.

- **Insert customer**

This procedure let people register their account as customer of the ordering system. To do that they have to insert their details, once that is done they can start using mobile application.

- **Recover customer's password**

In order to recover the password, user have to insert their email address and login. It will trigger the scenario of generating new temporary password for this user and will send it out to their email address. The procedure of sending emails from web application uses SMTP protocol, by default it is linked with gmail account. The settings are stored statically in configuration file **/config/autoload/email.local.php*. If login and email exists in the database it will save new password.

- **Food insert**
This procedure takes all food information, generates new identification number and returns it.
- **Food update**
In order to update the food, we are not overriding the record, but creating a new one with unique id and setting the original '*active*' column to *false*. This technique let us to keep track of all changes and prevent updates on the old orders.
- **Delete Food**
If user requires to remove the food, he is using this procedure. It just sets '*active*' column to false and it will remove current food from list. But in case there was any orders with this food it will save a link and old details.
- **Create order**
This procedure will create an order in case it receives all valid information and returns new generated id or null which mean something is invalid.
- **Food Order Insert**
When order is created and id returned, it starts filling it with products.
- **Order make payment**
This procedure compares the balance on the account and if it is sufficient to make a payment it will deducts the money from user and mark the order as paid. These operations are separate to prevent incompleteness of the transaction.
- **Order price**
This procedure calculates sum of order items and returns it.
- **Order cancel payment**
To cancel order payment we have to calculate sum of item prices and return them to customers balance.
- **Create voucher**
This function generates new voucher and places it in database.
- **Topup customer account**
Current transaction marks the voucher with customers account identification and adds money to their balance. After that this id is no longer valid for next transaction.

5.1.3 Tables

Order System

Ionic application users table:

Column	Type
id	binary(16)
gnumber	int(11)
email	varchar(40)
password	char(41)
cash	decimal(13,2)
active	tinyint(1)
created	datetime
updated	timestamp
name	varchar(40)
surname	varchar(40)
address	text
mobile	varchar(10)
cheat	int(10)

Table 5.1: Customers table

Vouchers table:

Column	Type
id	varchar(16)
amount	decimal(13,2)
used	tinyint(1)
customer	binary(16)
activated	datetime
created	datetime
modified	timestamp

Table 5.2: Vouchers table

Orders table:

Column	Type
id	varchar(16)
customer	binary(16)
comments	medium text
paid	tinyint(1)
collectionTime	time
collectionDate	date
created	datetime
modified	timestamp

Table 5.3: Orders table

Food table:

Column	Type
id	bigint(20)
name	varchar(40)
description	medium text
price	decimal(13,2)
type	enum('drink', 'roll', 'crisps', 'deal', 'misc')
picture	varchar(128)
active	tinyint(1)
created	datetime
modified	timestamp

Table 5.4: Food table

Food ordered table:

Column	Type
order_id	varchar(16)
food_id	bigint(20)
count	smallint(5) unsigned
created	datetime
modified	timestamp

Table 5.5: Food Ordered table

Administration

Web application users table:

Column	Type
id	unsigned int(10)
email	varchar(100)
password	varchar(100)
status	enum('Y','N')
created	datetime
modified	timestamp

Table 5.6: Users table

Web application resource table:

Column	Type
id	unsigned int(10)
resource	varchar(100)

Table 5.7: Resource table

Web application permission table:

Column	Type
id	unsigned int(10)
permission	varchar(100)
resource	unsigned int(10)

Table 5.8: Permission table

Web application role table:

Column	Type
id	unsigned int(10)
name	varchar(100)
status	enum('Active', 'Inactive')

Table 5.9: Role table

Web application role permissions table:

Column	Type
id	unsigned int(10)
role	unsigned int(10)
permission	unsigned int(10)

Table 5.10: Role permissions table

Web application user role table:

Column	Type
id	unsigned int(10)
user	unsigned int(10)
role	unsigned int(10)

Table 5.11: User role table

Management

Time available for ordering food table:

Column	Type
id	unsigned int(10)
weekday	tinyint(1)
open	time
close	time
active	enum('Y', 'N')
created	datetime
modified	timestamp

Table 5.12: Opening time table

This table stores information about available collection time:

Column	Type
id	unsigned int(10)
collection	time
active	enum('Y', 'N')
created	datetime
modified	timestamp

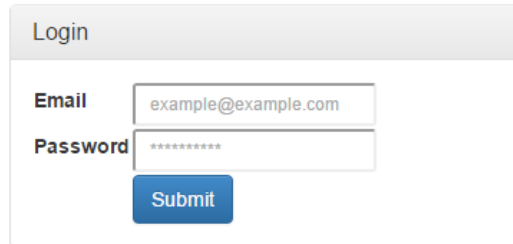
Table 5.13: Collection time table

5.2 System Administration and Management Web Application

In this section I will talk about functionality of this system. The amount of data that we are storing in our database is large and this system can be extended with minimal changes in the database structure. As you already seen what information we store in the system you can determine that customers details are stored separately from system users.

5.2.1 Admin Authentication

To authenticate users I am using ZF2 Auth ACL module. It is an open source module which can be accessed on GitHub. It contains 5 tables that describes website users, roles and route permissions. To improve systems security I made several changes into this module. [20]



A login form titled "Login" with a light gray header. Below the header, there are two input fields: "Email" with the value "example@example.com" and "Password" with masked characters "*****". A blue "Submit" button is located below the password field.

Figure 5.1: Login container

5.2.2 Voucher system

This system required operations with money and we come up with great solution, to implement voucher system. Customers have to buy vouchers to top up their balance on the account. Cards has to be printable and in order to do so I am generating a **.pdf* file that can be printed or saved for future printing. Voucher contains short information about company and *QR code* that allow to scan by smartphone to simplify code inserting process. Figure 5.2 shows an example of printed voucher.



Figure 5.2: Voucher example

To design this voucher and make it printable from web browser, I am using *DOM pdf* generator. It allows to display remote pictures and specify fixed positions for text. *QR Code* generating module submits the *HTTP*

request to Google API and receives a dynamic link to an image, which later in page generation I am using to add into *pdf* file.

QR code generator

QR codes are very popular type of two-dimensional barcode [21]. This module was used for generating QR codes for vouchers. When voucher is printed users can scan this code and top up their balance. This module is overridden for Zend Framework 2 and it uses Google Developers QR code generation API. [22]

DOM PDF module

[23]

5.2.3 Ionic routes & functions

Authenticate customer

Recover customers password

Change customers password

Sign Up new customer

Top Up customer balance

Approve customer's order and purchase

Show customers history

Show available food

5.2.4 PDF make

[24]

5.3 Mobile App

Chapter 6

System Evaluation

6.1 Web App

6.2 Mobile App

Chapter 7

Conclusion

Bibliography

- [1] (2016) Zend Framework 2 Overview. <http://framework.zend.com/about/>. [Online; accessed 13-April-2016].
- [2] Y. F. G. M. Elhakeem and B. I. A. Barry. (2013) Developing a security model to protect websites from cross-site scripting attacks using zend framework application. <http://0-ieeeexplore.ieee.org.library.gmit.ie/stamp/stamp.jsp?tp=&arnumber=6634012&tag=1>. [Online; accessed 13-April-2016].
- [3] H. J. La and S. D. Kim. (2013) Balanced mvc architecture for developing service-based mobile applications. <http://0-ieeeexplore.ieee.org.library.gmit.ie/stamp/stamp.jsp?tp=&arnumber=6634012&tag=1>. [Online; accessed 13-April-2016].
- [4] O. Corporation. (2016) Top reasons to use mysql. <https://www.mysql.com/why-mysql/topreasons.html>. [Online; accessed 13-April-2016].
- [5] N. Adermann and J. Boggiano. (2016) Composer documentation. <https://getcomposer.org/doc/>. [Online; accessed 13-April-2016].
- [6] (2016 (accessed on 2016-04-17)) Github homepage. <https://github.com/>.
- [7]
- [8] (2016 (accessed on 2016-04-17)) Html mdn homepage. <https://developer.mozilla.org/en-US/docs/Web/HTML>.
- [9] (2016 (accessed on 2016-04-17)) Css mdn homepage. <https://developer.mozilla.org/en-US/docs/Web/CSS>.
- [10] (2016 (accessed on 2016-04-21)) Html/css codeschool path. <https://www.codeschool.com/learn/html-css>.
- [11] (2016 (accessed on 2016-04-21)) Html/css codeschool path. <https://www.codeschool.com/learn/javascript>.

- [12] (2016 (accessed on 2016-04-17)) Javascript mdn homepage. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
- [13] R. Connolly. (2015) The advantages of using javascript for full stack development with an emphasis on node.js. <http://www.ronanconnolly.ie/js-advantages-in-fullstack-dev>.
- [14] (2016 (accessed on 2016-04-17)) Angularjs homepage. <https://angularjs.org/>.
- [15] (2016 (accessed on 2016-04-21)) React homepage. <https://facebook.github.io/react/>.
- [16] (2016 (accessed on 2016-04-21)) Ember homepage. <http://emberjs.com/>.
- [17] (2016 (accessed on 2016-04-21)) The ionic show // episode 1. <https://www.youtube.com/watch?v=uJAWaE11Jf4e>.
- [18] (2016 (accessed on 2016-04-21)) John papa's angular style guide. <https://github.com/johnpapa/angular-styleguide>.
- [19] R. T. Fielding. (2000) Representational state transfer (rest). http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. [Online; accessed 13-April-2016].
- [20] <https://github.com/arvind2110/ZF2-Auth-ACL>. [Online; accessed 16-April-2016].
- [21] https://developers.google.com/chart/infographics/docs/qr_codes#overview. [Online; accessed 16-April-2016].
- [22] <https://packagist.org/packages/adminweb/qrcode-zf2-module>. [Online; accessed 16-April-2016].
- [23] <https://github.com/raykolbe/DOMPDFModule>. [Online; accessed 16-April-2016].
- [24] <http://pdfmake.org/index.html#/gettingstarted>. [Online; accessed 16-April-2016].