Contents

[Development Methodology 2](#_Toc406331204)

[Project Plan 3](#_Toc406331205)

[DESIGN and TECHNOLOGY details 5](#_Toc406331206)

[Current state of the project 7](#_Toc406331207)

[Conclusion 7](#_Toc406331208)

[References 8](#_Toc406331209)

## Development Methodology

In this section we are going to discuss the methodology of software development life circle (SDLC) we chose to use to develop this project. After initial research we decided for Prototyping SDLC.

**Prototyping SDLC, what is it basically?**

In this approach the development team implements a ‘sample’ which have only very limited functionality of proposed project and show it to the customer. Customer provides the suggestions of improvements and development team implements them. This circle then repeated itself numerous times until the project is fully functional, tested and accepted by customer. This approach differs from other methodologies by avoiding doing the ‘big design in advance’ followed by implementation, testing and deployment phase. The project is rather developed by mutating the prototype with numerous design, implementation testing and deployment phases until the final product is build ***[1]***.

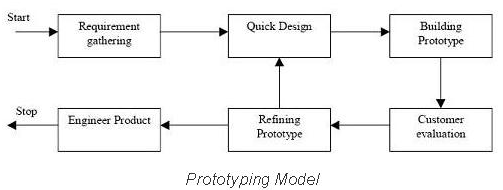


Figure Prototyping SDLC work flow

**Why did we decided for this approach?**

We decided for prototyping approach for various reasons

* We wanted to avoid ‘big design in advance’. To design web application in advance it requires an experience. Only experienced developers who already worked on projects with similar functionality and scope are able to do that. If ‘an amateur’ tries to design the application he will realize during the implementation phase that his design has flows and redesign is required. This will lead to lose of work hours and already done codebase. Basically you need an architect to design the house. We amateurs. We never build web application of this scope. With prototyping approach we be able to redesign the sample as many times as needed.
* We will learn with each prototype mutation iteration. In our case we first build simple but working sample build from prototypes. Data store prototype (relational database layer), data model prototype (objects to data mapping layer), controller prototype (business logic layer), and the view prototype (front end client layer). We make sure it all works together. Then we pick one proposed functionality and implement it whole way down through all the layers. And we learn from it. Implementation of next functionality will be easier and we get more productive over time. Hopefully after various iterations we will be able to call ourselves ‘the web developers’.
* Time restrictions. According to our project plan the first working prototype delivery is due to 14.February 2015. But we have been notified (by email 14.November 2014) that we must deliver the ‘working prototype’ due to 19. December 2014. According to our plan this is a two months ahead. Being still in research phase of project plan we assessed that only feasible approach under these circumstances is prototyping SLDC.

## Project Plan

In this section we are going to discuss how we planned to develop this project.

**What are we developing?**

Fantasy Hurling project is basically fantasy sport game. It’s rich web application not a static web page. As a player manages his fantasy hurling team a content dynamically changes. At the same time it must have a persistence capabilities. Basically a changes player made must be stored somehow and not get lost when player finishes to play. Data must outlive the application runtime.

Basically we will need to have three main layers.

* Front-End (code running in browser delivered by initial http request from web server)
* Back-End (code running on server listening and answering to http requests)
* Persistence Storage (relational database server listening and answering CRUD queries)

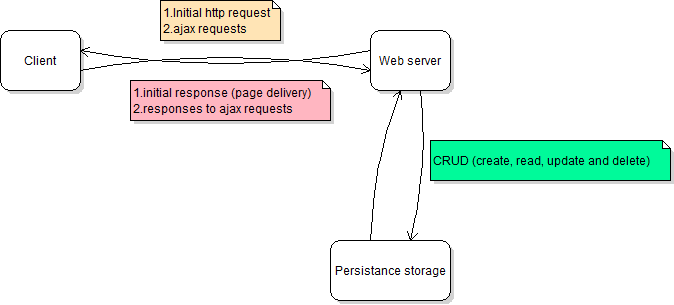


Figure Web Application general design

**Work Flow Plan**

Due to change of SDLC methodology all DEVELOPMENT and QUALITY ASSURANCE items are merged into PROTOTYPING ITERATIONS item.

All DESIGN items are reduced to PROTOTYPE DESIGNS (user interface prototype design etc)

**Who does what?**

**David Kelly**

* Front-End design ***responsibility*** (wire frames, photo shop, HTML5, CSS3)
* Documentation involvement
* Testing involvement

**Michael James**

* Front-End development ***responsibility*** (HTML5, CSS3, vanilla JavaScript, jQuery, jQuery UI)
* Documentation involvement
* Testing involvement

**Martin Zuber**

* Back-End ***responsibility***(MSSql database, .NET Web API 2, Entity Framework ORM mapper, Azure Deployment)
* Documentation involvement
* Testing involvement

## DESIGN and TECHNOLOGY details

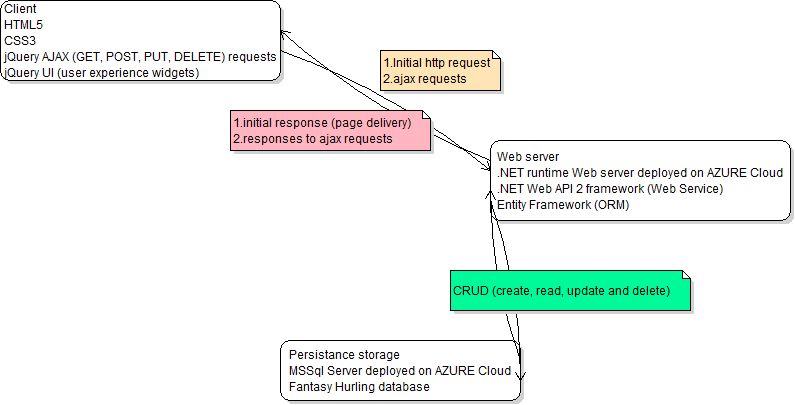


Figure Web Application in detail

**GIT and GitHub**

We are developing this application using Git distributed revision control system.

Fantasy hurling upstream repository is available at:

<https://github.com/Michaelcj10/Fantasy_Hurling>

Fantasy hurling repository clone is maintained and available at:

<https://github.com/zubidlo/group_project>

Web API prototype upstream repository is available at:

<https://github.com/zubidlo/FanHurApi>

These two repositories are going to get merged in later state of development.

**Front-End**

First front end prototype is simple web page using HTML5 ***[2]*** and CSS3 ***[3].***AJAX ***[4]*** request are made through simple web forms using jQuery ***[5].*** In prototyping iteration this web page will gradually turned to user friendly application using jQuery UI ***[6]*** widgets and methods.

Front-End development will be done using IDE. Some recommended options are Sublime Text 2 ***[7]*** or Aptana Studio ***[8].***

**Data Store**

We store Fantasy Hurling data in relational database tables. Initial prototype is design to mimic football like sport team and player statistics and it will gradually mutate into final ‘hurling’ state during prototyping iterations.

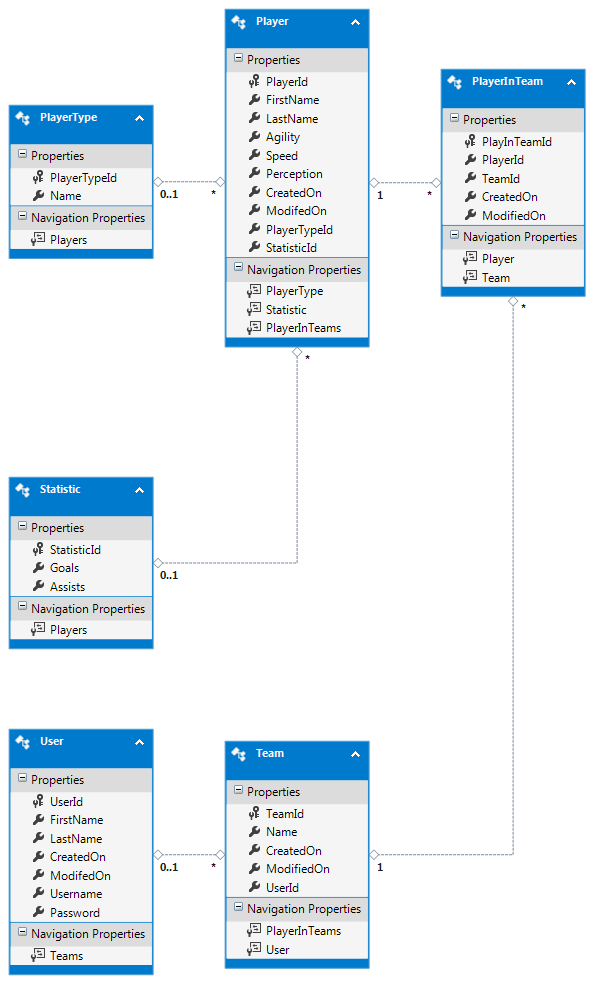


Figure Fantasy Hurling database prototype

Database will be deployed on Azure Data Store ***[9].*** At the end of development data store gets migrated to Azure Europe North Datacenter located in Dublin ***[10].***

**Back End**

Extensive research was done on web application back end development and we decided to build Web Api Service ***[11].*** Initial idea was to use Node.js framework ***[12]*** to build the API, but after additional research a decision was made to use ASP.NET Web API ***[13]*** technology. Some of the reasons for this change include:

* Complexity of task versus maturity of Node.js technology
* Potential lack of documentation if a problem in code arises
* Lack of quality connectors to relational databases, MongoDB (json based object database) is preferred with Node.js
* Additional frameworks to learn building Web API (express framework and more)
* Visual Studio 2013 Community edition free for use now
* 10 web servers free on Azure Cloud with Visual Studio installation
* Visual Studio fully implements Azure storage and web server deployment (on click in a menu and application and database is up on cloud)
* Comprehensive documentation and tutorials for all required frameworks and technologies.
* Technical support and huge community of developers if a coding problem arise.

I must add here that we expect a usage of additional tools and technologies during the project development as a need arises.

## Current state of the project

**What was done so far?**

* Web page prototype with working use case of AJAX CRUD request to Web API. Web page is maintained on localhosts only.
* Web API prototype capable of answering cross-origin requests ***[14]*** and generating SQL queries for requested CRUD operations. Web API prototype is deployed on Azure: <http://fantasyhurling.azurewebsites.net/>  
  API is in very early stage of development and it’s not optimized.
* Database prototype deployed on Azure Storage server
* This documentation.
* Project cooperators diaries.

## Conclusion

We choose not a trivial project. Being unexperienced developers we didn’t foresee the complexity and challenges involved in developing a fantasy sport web application. After realizing the scope of the project we changed our initial ideas how to approach this project development. We decided to follow prototyping methodology implementing and test one functionality at the time. Hoping to learn from each iteration and deliver a modern optimized rich user experience application.

In this occasion we decided to use the most modern technologies and designs. We are building Web API Service so that different clients could get access to CRUD methods. This way the additional clients can get developed easily. For example in the future we can easily implement Android client or Windows 8 metro client for our fantasy hurling application. We using Entity Framework ***[14]*** to automatically map database table rows to entity objects so we don’t need to write one sql query. We implementing our Web API using .NET Web API 2 framework which is powerful toolkit capable of asynchronous responses, cross-origin requests and much more. We are deploying the application up on Microsoft Azure Cloud.

This project is an opportunity for us to learn new technologies and get valuable firsthand experience and skills in web development.

## References

1. What is Prototype model- advantages, disadvantages and when to use it?. 2014. What is Prototype model- advantages, disadvantages and when to use it?. [ONLINE] Available at: <http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/>. [Accessed 14 December 2014].
2. HTML5. 2014. HTML5. [ONLINE] Available at: <http://www.w3.org/TR/html5/>. [Accessed 14 December 2014].
3. Introduction to CSS3. 2014. Introduction to CSS3. [ONLINE] Available at:<http://www.w3.org/TR/2001/WD-css3-roadmap-20010523/>. [Accessed 14 December 2014].
4. Ajax | MDN. 2014. Ajax | MDN. [ONLINE] Available at:<https://developer.mozilla.org/en-US/docs/AJAX>. [Accessed 14 December 2014].
5. jQuery. 2014. jQuery. [ONLINE] Available at: <http://jquery.com/>. [Accessed 14 December 2014].
6. Menu | jQuery UI. 2014. Menu | jQuery UI. [ONLINE] Available at:<http://jqueryui.com/m/>. [Accessed 14 December 2014].
7. Sublime Text: The text editor you'll fall in love with. 2014. Sublime Text: The text editor you'll fall in love with. [ONLINE] Available at:<http://www.sublimetext.com/>. [Accessed 14 December 2014].
8. Aptana . 2014. Aptana . [ONLINE] Available at: <http://www.aptana.com/>. [Accessed 14 December 2014].
9. Storage Documentation | Azure. 2014. Storage Documentation | Azure. [ONLINE] Available at: <http://azure.microsoft.com/en-us/documentation/services/storage/>. [Accessed 14 December 2014].
10. Azure Regions. 2014. Azure Regions. [ONLINE] Available at:<http://azure.microsoft.com/en-us/regions/>. [Accessed 14 December 2014].
11. Web API - Wikipedia, the free encyclopedia. 2014. Web API - Wikipedia, the free encyclopedia. [ONLINE] Available at:<http://en.wikipedia.org/wiki/Web_API>. [Accessed 14 December 2014].
12. node.js. 2014. node.js. [ONLINE] Available at: <http://nodejs.org/>. [Accessed 14 December 2014].
13. ASP.NET Web API | The ASP.NET Site. 2014. ASP.NET Web API | The ASP.NET Site. [ONLINE] Available at: <http://www.asp.net/web-api>. [Accessed 14 December 2014].
14. Cross-origin resource sharing - Wikipedia, the free encyclopedia. [ONLINE] Available at: <http://en.wikipedia.org/wiki/Cross-origin_resource_sharing>. [Accessed 14 December 2014].
15. Entity Framework. 2014. Entity Framework. [ONLINE] Available at:<http://msdn.microsoft.com/en-ie/data/ef.aspx>. [Accessed 14 December 2014].