Technology Discussion Document
version 1.3. September 2011

EDIT: A number of people have asked who we are and why we've created this document. Basically I am a developer turned entrepreneur who has created a number of small businesses over the years. My main website is www.inovica.com. I've created this document to share some of my thoughts for a project that I am involved in. Hope that it helps you or that you find at least something of use. Pay it forward :) Ade

Purpose of this document

I have recently been asked to help a company as they need to create a roadmap for the technology that they are going to use in the coming years. The idea is to investigate the technologies available, establish what is useful (and what is not) and we can ultimately create a specific framework to take forward. There is a huge change in the technology landscape at the moment and the problem that I have seen with some technology is that it is "cool" and we need to avoid this. We must only choose technologies which have a solid foundation and that will help us to create stable bases to run our businesses.

We would like you to have a read through this document, to join in discussions on the following forum that we have created for this purpose and to provide feedback on your thoughts:

xxxxxxxxxx

Overview

The following are some of the key areas of development that we need to look at, that we need help with and that we need to do some research on. Your thoughts would be welcome as I believe that only as a group and working as a team can we really benefit and make products which are truly awesome! :)

If there is a particular area that you are interested in exploring, please let me know as I want to split some research across the team.

Languages

We've been using PHP as our main programming language for many years now and I think its fair to say that PHP is the most widespread programming language for web development and from what I understand is used on about 75% of web servers. It also powers (at least at the front end) many popular sites, such as Facebook (though I understand Python and Erlang is used for the chat service). However, I think it is also fair to say that PHP has its issues and our experience is that other languages can be more powerful and easier to program in.

One thing we need to be careful of is that we don't choose a new language just because its the one the "cool kids" are using and we need to evaluate whether we stay with PHP or move to another language (or combine them, using the best features of more than one). Naturally we have a lot of experience with PHP and a lot of legacy code with it, so we need to bear that in mind. The languages that I would also like us to look at are:

- Python (which we are using anyway) (powers YouTube, much of Google)

- Ruby on Rails (which definitely IS the cool kids on the block)

- Javascript. There are people using this now at the server-level with good results (ie Twitter). As Yuri suggested we should look at Node.JS. He said that some of our apps that receive a lot of HTTP requests (such as EWS, RTS and VWB) would benefit from this as our requests will only grow as we increase our customer base. He has suggested that we maybe create a unified ssytem to receive these requests based on Node.JS

Naturally I'll entertain looking at other languages or opinions that people have about the above. There are ones such as Erlang and Haskell but I believe these are 'edge' cases where we would need them

Things to look at ...	
PHP offical site	http://www.php.net/
Ruby on Rails	http://rubyonrails.org/
Python official site	http://www.python.org/
Style guide for Python code	http://www.python.org/dev/peps/pep-0008/
Documenting Python code	http://www.python.org/dev/peps/pep-0257/
Erlang	http://www.erlang.org/
Haskell	http://www.haskell.org
Node.js	http://nodejs.org/

Templates and Frameworks

Currently with much of our code we are using PHP along with our home-grown template and framework system. It has been a great system and has served us well, but I feel that there *may* be other systems that are more flexible, or we may look at improving our template system itself. Now, frameworks seem to be defended with an almost religious zeal, so I am outlining only a few here. The purpose of using a common framework is that all developers will be working with the same system therefore some of the issues that we have had will be averted.

The following cover what I believe we need here:

- Universal way of working that every member of the team can use and understand
- A template system that is very easy for a *designer* to use
- A framework system that when updated does not mean reworking previous code (if possible)
- A template system that is secure. I do not believe we should allow any code to be run from within a template

Naturally the choice of template may, in part, be governed by the programming language(s) that we ultimately use, although I have found some template systems run across some core languages.

Template systems to look at...	
Liquid (Ruby)	http://www.liquidmarkup.org/
Mustache (cross language templates)	http://mustache.github.com/
Mako Templates	http://www.makotemplates.org/
Mustache Tutorial	http://net.tutsplus.com/tutorials/javascript-ajax/quick-tip-using-the-mustache-template-library/
Good blog post about Mustache	http://blog.couchbase.com/mustache-js
Good read about how Mustache templates are logicless	http://www.quora.com/What-are-the-benefits-of-logic-less-templates-like-Mustache
The logic behind Twitter (which uses Mustache)	http://engineering.twitter.com/2010/09/tech-behind-new-twittercom.html
CTemplate (which Google uses for their C++ stuff). This inspired Mustache	http://code.google.com/p/google-ctemplate/
Teng - template Engine for PHP, Python and C++	http://teng.sourceforge.net/
Themeforest. A site selling templates in various template languages	http://themeforest.net/item/network-wp/69831
ctpp - cross-platform templating that is written in C++	http://ctpp.havoc.ru/en/index.html

Frameworks to look at...	
Pyramid (Framework for Python)	https://www.pylonsproject.org/
Django (Framework for Python)	https://www.djangoproject.com/
Flask (Microframework for Python)	http://flask.pocoo.org/
Cake (Framework for PHP)	http://cakephp.org/
Ruby on Rails (Ruby framework)	http://rubyonrails.org/
Symfony (PHP Framework)	http://www.symfony-project.org/
Sproutcore (Javascript Framework)	http://www.sproutcore.com/about/
HTML 5 and CSS Javascript library	http://www.modernizr.com/
Prototype JS Framework	http://prototypejs.org/
Google Closure library	http://en.wikipedia.org/wiki/Google_Closure_Tools

Zend Framework	http://framework.zend.com/
Naturally there's loads more...	

Databases

Regarding databases we have traditionally been focused on using MySQL. There is a strong push these days towards NoSQL, so I think we need to evaluate the benefits of this approach as well as looking at other SQL-based tools such as Postgres which gets a lot of praise. We also need to look at how our systems interact with databases. Please have a read of the following:

Things to look at ...	
Look at ORM (object relational mappers) - Thanks Lucas	http://www.doctrine-project.org/
SQL Alchemy (SQL toolkit and ORM for python)	http://www.sqlalchemy.org/
S3 Tools for working with Amazon S3	http://s3tools.org/s3tools
Cassandra - Scalable, distributed database - looks cool	http://cassandra.apache.org/
Mysql master/slave	http://blog.dotcloud.com/high-availability-mysql-masterslave-now-on-do
Kyoto Cabinet	http://fallabs.com/kyotocabinet/

Caching & Search

The caching of data and pages can reduce the load on the servers, improve the load time and therefore improve the users experience. A traditional database, with a traditional structure, can quickly cause problems for search, especially as the database grows. Therefore we need to look at how search can be improved and how we can manage larger datasets.

Things to look at ...	
Squid	http://www.squid-cache.org
Varnish	https://www.varnish-cache.org/
Memcached	http://en.wikipedia.org/wiki/Memcached

Scaling PHP apps with Varnish (Article)	http://www.ibm.com/developerworks/opensource/library/os-php-varnish/index.html
Sphinx	http://sphinxsearch.com/
Lucene (indexing engine)	http://lucene.apache.org/solr/
API for Python for Solr (part of Lucene)	https://github.com/tow/sunburnt
Redis. Fast memory DB to store frequently accessed data	http://redis.io/
Article on Hadoop	http://tcfast.com/2011/07/17/hadoop-startups-where-open-source-meets-business-data/

API's

Nearly every new web app or product which comes onto the market these days has an API. It makes it easy for the owning company to provide a hook into their apps for other people to extend and naturally it is often a very good fit for developers also. My own feeling is that we need to have an API for everything that we do from now on and that we should utilise our own API to create our products. Doing it this way means that we know that the API works. My suggestion would be to have a private and a public API layer, for obvious reasons.

Rather than outlining all my current research on this here, I think it would be useful for you to read a couple of interesting discussions relating to APIs as well as some other links. My suggestion would be to read these, look at the links and then to feedback in the forum about it. From here we can then start to make some progress.

Things to look at...	
Why you absolutely MUST write an API when you write your next app	http://news.ycombinator.com/item?id=1931396
Best Books on API design. Some great links here and some good video links	http://news.ycombinator.com/item?id=1668561
Little manual of API design (discussion)	http://news.ycombinator.com/item?id=2796371
Little manual of API design (link) Great book on this as a PDF for you to download	http://dl.dropbox.com/u/5970491/api-design.pdf

Programming Methodology

Now that we are growing our team we must create a consistent way to program to ensure that everything is coded in the same way, making it easier for any member of the team to become involved. This includes:

- Documenting code. We need to adopt a standard here for this
- Documentation. All of our apps need documenation
- Version control. We need to use a version control system (such as Git or Mercurial, which we currently use)

- Deploying apps - also need to look at across multiple servers
- Bug tracking. We need to ensure we have a good bug tracking system. Currently we're using Assembla for this and version control
- Project management.
- Method. We need to look at 'how' we code. My suggestion is to look at Agile development
- Training. We need to ensure that each team member receives enough training in the areas they are concentrating. This either means the provision of resources such as books, online training or 1-1 training from other team members
- Testing. We need to do both functional testing as well as load testing on our apps

We need to have discussions on the above as well as someone to take areas of these forward and make decisions on them

Things to look at...	
Assembla	http://www.assembla.com
Github	https://github.com/
Read The Docs - online documentation site (need to find out if it allows commercial work)	http://readthedocs.org/
Selenium automates browsers. Good for automated web testing	*http://seleniumhq.org/*
Load testing	https://secure.blitz.io/pricing
Unit Testing	http://en.wikipedia.org/wiki/Unit_testing
Test Driven Development	http://en.wikipedia.org/wiki/Test-driven_development
Agile Development	http://en.wikipedia.org/wiki/Agile_software_development
Deploying apps using Chef (article)	http://help.opscode.com/kb/otherhelp/build-a-django-stack
Literate Programming	http://en.wikipedia.org/wiki/Literate_programming

Design

This is one area where I have the least knowledge, but I wanted to put it into this document so that we can come up with a 'plan' for how we are doing our design. It links to our 'template' side of things

Design of websites has changed hugely over the years - both in terms of look as well as how these sites work. We now have the requirement for more interaction on a site (Ajax etc) as well as multiple browsers and platforms (desktop, laptop and mobile devices). We need to look at the best

way of designing now to ensure we're as flexible as possible and as current as we can be. This will take some research, but my feeling is that we need to bring a designer in who designs using modern methods - currently we have struggled in terms of time for experimentation and I believe that the above will be the fastest way to do it. A designer these days can't just 'design' and they need to have enough technical competence to be able to understand how the front-end works, especially using Ajax

Things to look at...	
Susy (CSS design stuff). Works with older browsers also	http://susy.oddbird.net
Javascript library (could also be under frameworks)	http://script.aculo.us/
Backbone	http://documentcloud.github.com/backbone/

Servers, management, monitoring etc

In terms of servers traditionally we have just rented dedicated servers. Over the past few years we have started utilising the Amazon 'cloud' with their Amazon Web Services. This is a really useful service for scaling quickly up and down, but it can be expensive if not watched carefully. What we need to do, I believe, is look at a hybrid system with some dedicated servers as well as utilising a cloud service for scaling. Some key aspects to look at here is:

- Easy mirroring of servers and server data both within a data centre as well as between data centres (for redundancy)
- Management of these servers. We need to look at getting a sysadmin to manage the security and stability of our servers as well as the ongoing management
- We need to look at monitoring software to monitor the applications as well as the server and services on each server.
- We need to look at how to create applications which scale

Things to look at...	
Distributed server monitoring	http://www.zabbix.com/
Server Density monitoring	http://www.serverdensity.com/
getting around an AWS failure:	http://news.ycombinator.com/item?id=2477877
Server monitoring (Spotify use them)	http://newrelic.com/

Protecting us - Scaling, Distribution, backups and Failover

Historically we have created applications where one server would be more than enough. As our

number of users grow, and the type of applications that we create become more complex we have a need to ensure that our applications can scale well and that if there is an outage or problem with a server/provider that the application will continue to run from a backup system. In order to do this we need to look at the following:

- Failover. Our applications must be able to run across different providers. We need systems that will automatically failover but also systems that will ensure the consistency of data across machines. I would like us to look at different failover systems, including DNS Failover.

- Backup. This is taking snapshots of the data that we have, on a regular basis, and storing them somewhere.

- Scaling. Our applications need to be able to scale up (and down) easily so that if an application starts to receive high load that another server will be used to start to take some of the load. Utilising some services I understand that it is possible to automatically scale (such as using Amazon AWS) and we need to look at this.

Things to look at...	
Reliability, Availability and Scale (article)	http://afeinberg.github.com/2011/06/25/reliability-availability-scale-interlude.html
Static sites to improve load	http://highscalability.com/blog/2011/8/22/strategy-run-a-scalable-available-and-cheap-static-site-on-s.html

Company-specific Technology

After creating this document I think there are only a couple of items that are specific to this company

- VXML. We need a greater knowledge of VXML. Someone needs to learn about it
- Twilio.com. This looks really useful for creating web/telecoms apps
- Phono.com. Similar to Twilio
- Automated Dialing. We need to find a method of automated dialing so that we can test (and stress test) our applications
- Scrapers. We need to find the best way to scrape data as well as to create a tool which can work similar to a macro recorder, such as how Mozenda works
- Sitebuilders. We need to have a method of creating sitebuilders, compelte with the ability to switch templates easily. Code should be centralised and shared, so that any change will be spread across all sites

Things to look at...	
Scrapy - useful scraping framework for Python	http://scrapy.org/
Beautiful Soup - scraping framework for Python	http://www.crummy.com/software/BeautifulSoup/
Twilio	http://www.twilio.com
Phono	http://www.phono.com

VoiceXML	http://en.wikipedia.org/wiki/VoiceXML

Other technologies

The following are some links and technologies that look interesting and worth looking at:

Python module for async processing automation	http://celeryproject.org
messaging broker backend - used to exchange of information between the app(s)	http://www.rabbitmq.com
Programmable email	http://mailgun.net/
Python Requests	http://docs.python-requests.org/en/latest/index.html
Web Sockets	http://en.wikipedia.org/wiki/WebSocket
Asynchronous Programming and Twisted (python) and discussion	http://news.ycombinator.com/item?id=2904539
Not a technology, but a useful helpdesk app	http://www.sproutit.com/
Historious Blog - some interesting stuff about how they built their system	http://blog.historio.us/

Sites / Links to look at

Here are a few sites that make good, and useful, reading:

High scalability	http://highscalability.com
Hacker News	http://news.ycombinator.com
Hacker News Discussion on technologies people worked on in 2010	http://news.ycombinator.com/item?id=2053956

What have you found that is useful?

If you've read this far and want to comment (even if its to tell me what an idiot I am!!), feel free to email me at adrian.teasdale@gmail.com or comment on Hacker News about technologies that you have found useful:

http://news.ycombinator.com/item?id=3025913

We are looking for at least one full time developer and a great UI designer to join us, even remotely