Important Topics in Software Strategy For Business

Platform Choices - An Executive Overview

www.workflowproducts.com
Workflow Products, LLC
7813 Harwood Road
North Richland Hills, TX 76180
817-503-9545

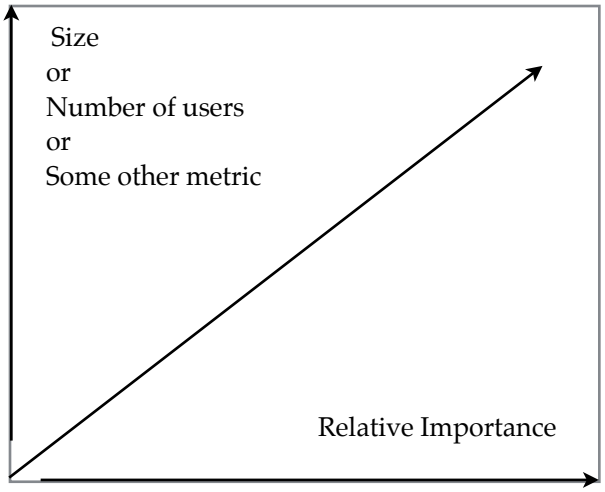
Table of Contents

Assessing the software landscape in an organization

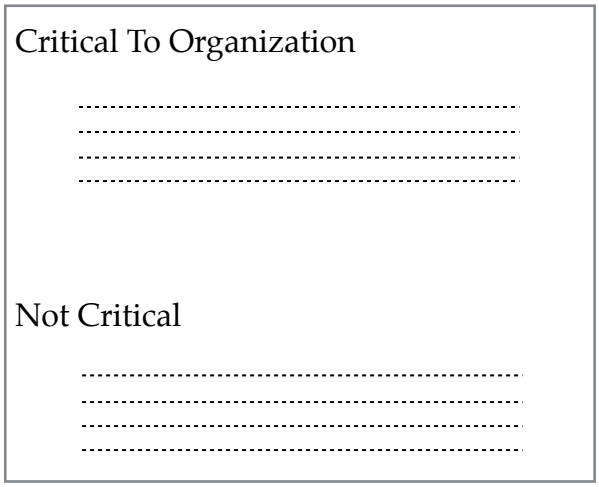
When looking for activities that should be handled by an application, we look for just one thing. **Documents that are updated regularly.** Any document that is updated regularly is not a simple document. Its a document that is being used as an application.

You may be wish to evaluate a document that is updated regularly as not critical to the organization. This is normal. A document that is updated frequently may in fact be unnecessary because it is used to manage some trivial activity. A judgement should be made, typically by management. **The judgement should be based solely on the importance of the activity.** Frequently, organizations will try to use other metrics to gauge importance and as a consequence, whether the activity should be brought into an application. This is discouraged and will lead to wasted effort.

If you have two activities and one is managed by the whole company and the other by one department, it can easily seem as if the former is more important. **In reality, every system either meets the bar of "critical" or it doesn't.** As a result, its best to have management class activities as either "critical" or "not critical".



Less Effective



More Effective

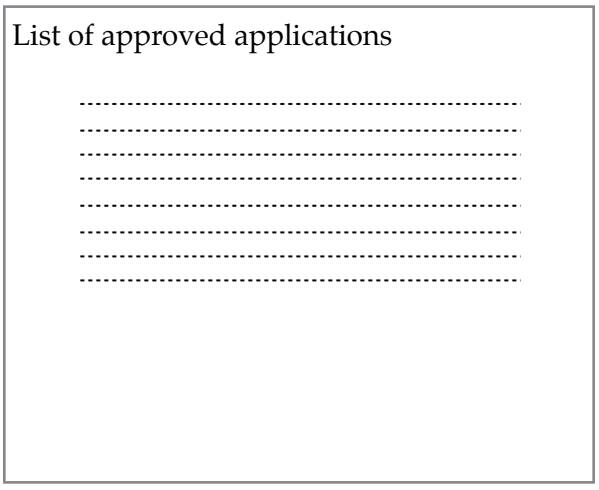
The Evolution Of Applications

New activities are introduced into an organization due to change. For the purposes of software management, whether the source of change is internal or external is not important. What is important is to acknowledge that change will occur and with it new needs will be created and fulfilled. **If your organization has a plan then the solutions will grow like a well kept garden.**

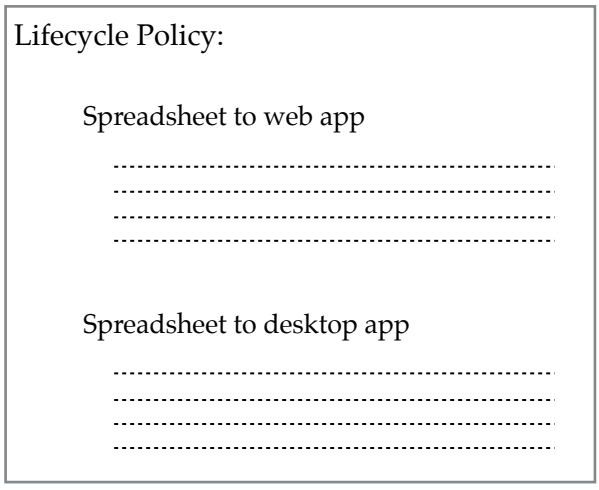
Management in collaboration with IT should take the time to choose application lifecycle options for use by non-IT staff. Additionally, security and backup policy should be posted prominently. A clear policy stating where documents must be stored will prevent document loss.

All organizations will have a changing landscape of skill sets. Significant risk and waste can be avoided with a "lifecycle policy". **A lifecycle policy is a simple document that enumerates organization policy for application activities.** With approval, documents can be used as applications for non-critical activities. Other activities will start as a document and be reviewed periodically to determine if its time for the activity to be moved to the organization's chosen application platform.

For example, an employee is given a critical task which is new for the organization. The employee consults the Lifecycle Policy and possibly the IT department. As a result, a spreadsheet is designed. In the beginning, the design may not be completely suitable. In the first weeks it is improved. Once the design is working and the alterations have slowed then its time to reevaluate. Is the need still critical? If so, resources are applied and the spreadsheet is upgraded to a simple application. The app is now part of the organization's application suite and can take advantage of the organizations chosen application platform(s).



Minimum



Recommended

Application Platform Types

Applications can be classified into three broad categories: desktop, web and mobile. The features and differences are numerous. Choosing your platform carefully will pay big dividends down the road. We will start with some broadly accurate generalizations.
Desktop applications typically provide for large screens, mice and a posture that means business.

Desktop Applications

With all these advantages, it's no wonder that when you need the richest possible user experience, you'll want a desktop application.
Desktop applications are typically unable to run on tablets, phones or other devices, in fact, they are often bound to a particular operating system such as Microsoft Windows. In many situations, this isn't a limitation and in those situations, the lack of compatibility actually manifests as a strength (consistency of experience). Users have come to expect rich experiences from desktop apps such as rich copy/paste functionality, drag and drop, full printing options and even import and export functionality.
Desktop application development is more robust over time due to a less complex architecture and less tools needed. A desktop application development tool will also typically have an upgrade path whereas typically web technologies do not.
Desktop apps can range from low to very high cost depending primarily on platform type. The next section is an overview of platform types.
Web apps do not need to be installed and require less support than desktop applications. However,

Web Applications

user interface tasks such as copy/paste, drag and drop, printing and other functionality is typically inferior. This inferiority is not inherent to the web or browser limitations. It simply reflects the current state of the art. Web applications have until very recently been limited to a primitive user interface toolkit by browser vendors. Developers have had to exert great effort to provide rich interfaces and that is expensive. Technology is advancing in this area and desktop applications have seen their advantage begin to shrink. By 2020 a significant number of web applications will reach parity and that should drive the remaining web applications to improve.
If your users are remote, a website can be lot easier to support. Sometimes your user profile dictates that you simply have to build a website. If you need to interface with customers directly, then a website may be your only option since remote support for a desktop application could quickly become prohibitively expensive.

Mobile Applications

Mobile Applications are a category unto themselves. They can be classified into three types; native, web and hybrid. There are a large number of mobile platforms and there are many considerations when evaluating. We will again make some broadly accurate generalizations.

Native applications offer the richest experience on a mobile device.
The better experience costs more because native applications are more time consuming to write. In order to get a native application for each mobile device type you can use one of the many cross platform development kits available. These paid software programs allow you to develop one app and release it to multiple device types at once. The rich experience of native applications covers a wide area of application types and a typical business application does not utilize even a small fraction of what native applications were designed to provide. As a result, native applications are most often chosen by public corporations or organizations with regulatory requirements such as the Sarbanes–Oxley Act, ITAR, HIPPA etc...
Native apps typically cost more to get from start to first release than other types of applications. Developer account costs on mobile platforms are trivial, but some vendors require your app to go through an approval process before release. Unexpected delays and rejections make native apps riskier than most other app types.

Web applications can be written to work for desktop and mobile.
For typical businesses, a web application can generally be written to work well regardless of what device people are using. Often, the website will be written to serve desktop users first, then selected parts of the desktop application will be re-written for mobile devices. This helps keep development time and cost down.
Generally web apps are inexpensive to start, but get more expensive to update over time. Web apps are complex, with many moving pieces. As the app expands, maintenance, security and performance become more of an issue. Many web tools go out of fashion and finding appropriate developers can increase cost.

Hybrid applications utilize a native shell to display a web application.
Hybrid applications are used to add some native functionality without increasing the cost of the solution to the level of a native application. There are many toolkits available that allow a developer to write a single app and deploy to several platforms. An organization would not typically use a hybrid app for a business oriented application. Often a hybrid app is used for public facing activities such as branding and other marketing activities.
Hybrid apps approach native apps in cost, which makes sense since they require both a web app component and a native app component. The benefit is reaching more platforms from one effort. It can be expensive to find replacement developers for some toolkit types. Hybrid apps are essentially native apps making risks associated with the approval process the same.

Desktop Application Platforms

Note the green row. That's a sweet spot between the high risk and high cost solutions.

	Desktop Platform	Minimum Initial Cost	Ongoing Cost	Risks	
High Risk	Spreadsheet	low	low	data integrity, confidentiality, data loss, security	
High Risk	MS Access, Document Mode, Single User	low	moderate	confidentiality, data loss, security	
High Risk	MS Access, Document Mode, Mult-user	low	moderate	confidentiality, data loss, security, performance issues	
	MS Access, Application Mode, PostgreSQL	< $1000	moderate	low	Advanced Platforms
High Cost	MS Access, Database Mode, SQL Server	five figures	moderate	low	Advanced Platforms
High Cost	Visual Studio and SQL Server	six figures	high	low	Advanced Platforms
High Cost	Oracle, IBM DB2	seven figures	negotiated	low	Advanced Platforms
High Cost	SAP	eight figures	negotiated	low	Advanced Platforms

The above chart is short on detail and there are always exceptions but the overall profiles provided are based on very wide experience. "Minimum Initial Cost" numbers exclude hardware costs.

<u>High Risk</u>

Spreadsheets are extremely useful for virtually all businesses and we encourage you to use them. A spreadsheet (or database document) only becomes a liability when it begins to be used to manage some critical part of the business. At that point they become a liability. The function should be moved to an application to avoid data loss and other risks.

<u>High Cost</u>

Application platforms come in a wide array of price points but the differences are surprisingly few. Desktop platform choices that use MS SQL Server operate at a higher ongoing cost than PostgreSQL or other advanced database systems due to the MS T-SQL procedural language. T-SQL requires more lines of code and ambiguous constructs, for example, the IF construct. Consequently, T-SQL adds significant overhead to developer productivity. PostgreSQL is particularly notable in offering multiple procedural languages. This can cut costs significantly when a particular problem benefits from a specific language. Some of the Advanced Platforms have low cost or free versions that remove the security features. As a result we don't recommended them for business applications.

Evaluating Desktop Application Platforms

There are many different ways to evaluate platforms and a comprehensive comparison is beyond the scope of this document. For our purposes, the important considerations for each platform should help you get started. If you need further explanation please contact us.

Documents and Spreadsheet Tasks Versus Application Tasks

Recommended Use Cases	Replace With An Application
• Single-use data gathering and reporting. • Storing historical data for infrequent use. • One time reports. • Report prototyping. • One-off presentations and charting. • Reference or work instructions. • Records of correspondence. • Data that needs to be sent to one person. • Data that is edited by one person.	• Reports are created regularly. • Data is being transferred from one document or application to another document or application. • Data is being entered or updated regularly (daily, weekly or monthly). • Data is for use by several people. • Data is updated by several people. • You've experienced errors in data entry. • Data has been lost.

Note that solutions created in documents and spreadsheets are often created by non-IT professionals. **Managers, analysts, and administrators create these solutions without budgets or guidance as part of their job. These solutions would not make economic sense if IT professionals fulfilled them.** Although solutions created by non-IT professionals might suffer from poor design, users should not be discouraged from fulfilling their own ad-hoc and short term needs. Instead, IT should support users by providing assistance such as help desk and training resources.

MS Access Document Mode

Using Microsoft Access in Document Mode is not recommended. Over the years Microsoft has added so many features to Access that investments in document mode can lead to some difficulty when it's time to upgrade. In order to avoid that, it's recommended that developers have experience using MS Access with a database. If you have a proper developer, then its only pennies more to move on to Application Mode. In light of the small difference in cost but large difference in benefits, Document Mode is not recommended.

You may encounter a developer who wants to create a database in Document Mode as a first step toward Application Mode. This is very useful and can help keep costs down.

Evaluating Desktop Application Platforms

MS Access and PostgreSQL Versus Advanced Platforms

- Make data entry easier/faster.
- Validate data upon entry.
- Eliminate double entry.
- Create reports on demand from a single click.
- Display real-time metrics.
- Create, store and retrieve documents.
- Automatically distribute documents on demand or on schedule via email.
- Prompt employees to action.
- Any data-driven application, such as accounting software, can be recreated and then customized for your company.

Recommended Use Cases

- Data is covered by regulatory or other considerations such as the Sarbanes–Oxley Act, ITAR, HIPPA etc...
- Due to regulatory or other considerations, Employees must be treated as untrusted users.
- The organization is a public company and regulatory or other considerations require data to be treated with special requirements.

Possibly Time To Upgrade

MS Access' popularity can be attributed to its speed, low cost and versatility. Professional developers can use Access to create very sophisticated solutions with maximum return on investment.

MS Access solutions require significantly less code, but more importantly, partially complete applications are always able to be run, tested and released to production. As a result a developer can create solutions more quickly than when using an advanced platform. In fact, advanced platform developers sometimes use MS Access for prototyping.

MS Access is sold with Microsoft Office but you don't need to purchase MS Access to use it for your organization. Only the developer is required to purchase a license. Meanwhile, MS Access solutions can easily create Word documents or Excel spreadsheets at the click of a button. MS Access apps are as rich and easy to use as any other application. Using Microsoft Access to create an application often causes lots of rich behavior to be included without the developer having to do any additional work. This keeps costs down and offers users a higher quality experience.

If you have special requirements and you'd like a second opinion on whether it's necessary to upgrade to an advanced platform, contact us for a consult.

Answering Objections From Developers Of Other Advanced Platforms

Visual Studio, Microsoft SQL Server and other advanced platforms are all very expensive. If you're considering these platforms it should be because you have problems that advanced platforms were designed to solve. Obviously, if it isn't necessary to use an advanced platform in your situation then it makes good fiscal sense to avoid it. If your organization has policies in place that limit your options, start by obtaining documentation. If you're in a position to obtain an exception it can be helpful to have a technical advocate on hand. Try to be aware of the qualifications and motivations of all actors.

Not every developer is sufficiently experienced in Microsoft Access to answer all questions. One can use MS Access for many years without ever becoming knowledgable about Application Mode. Consequently, a person can feel qualified to give information when in fact they are objectively unable to do so.

Developers that work with other advanced platforms may have insufficient experience with MS Access. It often happens that developers use Document Mode as a stepping stone to an advanced platform. Despite their advanced skills they may still be uninformed about Application Mode.

It can be harmful for a developer to spend significant time working with a different platform. Despite the lower productivity and higher cost, once a developer has become skilled in an advanced platform there are strong reasons not to 'go backwards' and use MS Access for a project. Advanced platforms require special skills acquired through lots of discipline and experience. Some platforms will routinely require developers to be skilled at a dozen technologies or more. Due to the time it can take to acquire all the skills necessary to produce even a rudimentary application, these skills can command high prices. A developer will have very strong motivations to present options that utilize the skills they have and no more. A second opinion could save you five figures or more.

If a consultant is equipped to serve $200K+ solutions, they likely cannot profitably manage opportunities worth far less than that. When consulting with outside contractors as to whether requirements can be met by Microsoft Access and PostgreSQL, be sure to consult with an appropriate contractor. If the organization supports other advanced platforms at all then they are unlikely to provide a quote for Microsoft Access and PostgreSQL at all.

The overall theme here is caution. Be thorough in vetting consultants for appropriate experience and knowledge. No one has to be malicious or deliberately hide information to not supply you with all your available options. Since there is so much money at stake its worth a few extra phone calls.

The Sweet Spot
MS Access and PostgreSQL

As a desktop platform

Microsoft Access is the most popular database software in the world and has been for over a decade. Connecting Microsoft Access to a database removes risk of data loss and performance issues. The differences are driven by connecting MS Access to a database. Once MS Access is connected to a high performance database, we essentially convert a MS Access document into a MS Access application.

A Microsoft Access app is not always 'locked down' like a typical desktop application. It is an optional feature of MS Access that a developer can 'lock down' a MS Access app in order to prevent users from getting under the hood of an application and looking around.

In practice, the reason a user may attempt to do so would be to create and run their own queries or reports. If they are profitable, a developer can then add these queries and reports to the distributed application. No other form of advanced platform offers this level of productivity. If you have even one user who is trained in MS Access the benefits can be substantial.

MS Access' chief advantage is in the low cost of ongoing development. Changes can be made throughout a working day and users need only close and re-open the application to get new features and bug fixes. Only web apps offer a faster upgrade process. Most other platforms require the user to install the updated version of the program.

The PostgreSQL security model is exceptional and has a better track record than MS SQL Server. Initial cost approaches zero while ongoing development cost is low compared to MS SQL Server and about the same as other advanced platforms for typical business apps.

PostgreSQL certainly excels in certain domains such as geographic information systems (GIS). The easy programmability of PostgreSQL and its open source nature have, over the years, attracted domain experts in data heavy fields such as statistics and bioinformatics. These people have in turn contributed software to the community, for example, PostGIS, that frequently has no equal in other advanced platforms.

As a rapid prototype platform

A few developers that program in advanced platforms will use MS Access in the prototyping phase of development. This is generally a strategy used by large companies with in-house development resources that wish to keep development costs down. Public corporations or organizations with regulatory requirements such as the Sarbanes–Oxley Act, ITAR or HIPPA may have a specific need to limit the distribution of software to a particular platform.

Evaluating Web Application Platforms

A quick search of Wikipedia will result in over a dozen common languages and over a hundred web 'frameworks'.

https://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

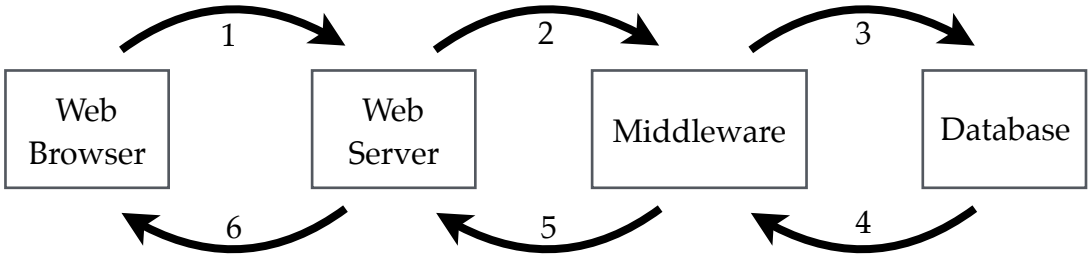
The above list is not comprehensive but instead reflects the platforms chosen by Wikipedia editors. It also doesn't give any indication of popularity. Less popular platforms don't offer a sufficient pool of experienced developers and are not recommended.

Up until very recently there was no objective way to evaluate web platforms. This is because all web application design is based on the same architecture regardless of platform, framework or language.

A web application has four parts; the web browser, the web server, the middleware and the database. Developers can put code in all four places but usually just the Middleware.

In the Wikipedia article you'll see many platforms and frameworks listed. All of them refer to code that goes in the box labeled 'Middleware' below.

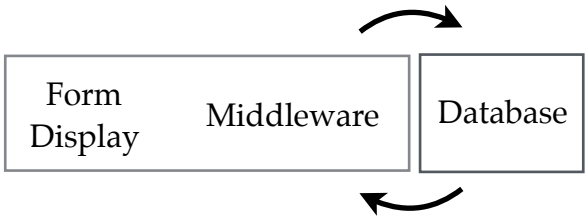
When a request is made by the user at the web browser, the web browser starts a chain of events that follows the arrows below.



For our purposes it would help if we name this chain of events "the request-response cycle".

Web Application Platforms(continued)

The web application request-response cycle is much more complex than the desktop request-response cycle. For comparison, here is an example of a desktop application request-response cycle.



When a request is made by the user, the form consults its own code. If it needs to contact the database it does so. Note that the form display and middleware are tightly integrated; to the point where its appropriate to understand them as one entity. On some platforms the database is also tightly integrated to the rest of the platform.

The difference is clear. The next question is "Why do we need so many pieces to make a web app?".

Web Browser

The reason we need Javascript is because its all we have in the web browser to enable rich experiences. Other technologies were produced to compete with Javascript. They used the browser plugin architecture. If the user wanted to view "rich" content they had to download software into their browser to enable it. Unfortunately, the developers of these rich experiences tied their technology to large screens, keyboards and mice. As a consequence, they fell out of favor once the smartphone was born.

Web Server

The reason we need the web server is because web applications have domain names. Web servers give you the ability to serve more than one domain name on the same server. Although they constitute a net increase in the number of pieces used to create our application they don't typically increase complexity. Web servers are worth the effort for the added capabilities they offer.

Middleware

The reason we need the middleware is because web browsers can't talk directly to the database.

Database

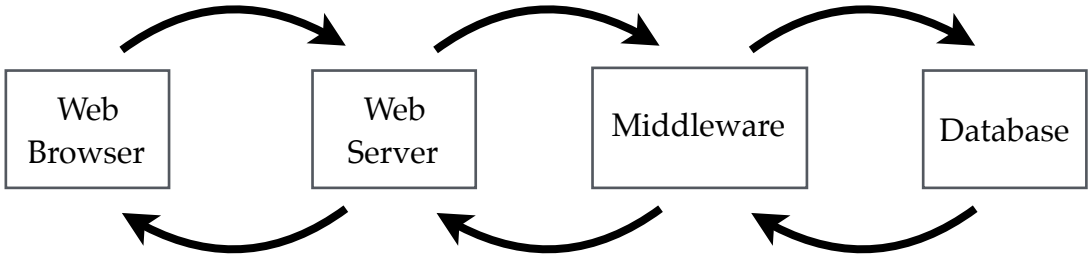
The reason we need the database is because databases are the only safe way to store and get at data quickly. Data safety is important but if a web application is slow it will be difficult to use. Databases are the fastest way to get at or summarize large quantities of data.

Web Application Platforms(continued)

Lets look at a simple example of a typical web application using Middleware.

In our example, an organization wants a web app to display a combo box, otherwise known as a drop down box. When the user clicks the drop down it should display a list of choices from a table in the database.

The Old Middleware Paradigm




```
<script>
// lots of code here
</script>
<body>
  <div id="combo"></div>
</body>
```

To make this happen under the old paradigm, the developer has to write code in several places. Here we show the HTML page. We've shown where the code goes in the script. Script code is Javascript.

Note that the combo box is just a "div". In a page with a lot of items on display there will be a lot of DIVs to draw them all. As HTML documents grow it takes more time to alter them because the DIVs make it difficult to read the page.

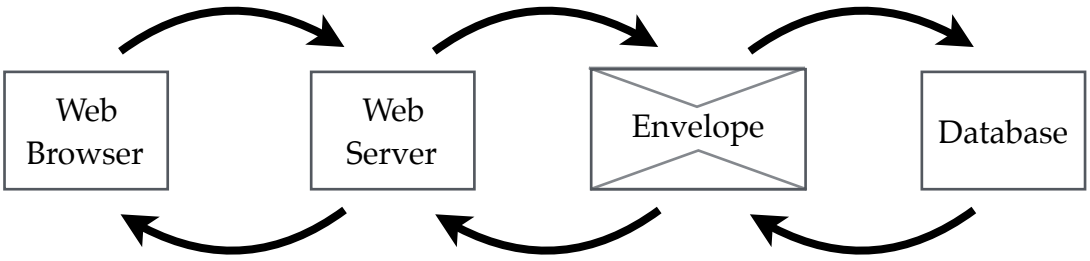
The developer is far from done. Once the code in the HTML page is complete, only the request and display of the data has been accomplished.

In the middleware we now need to write code to accept the request, talk to the database, make sure we have permission to see the data, fetch the data and then return it to the web page in a format that our HTML code can accept.

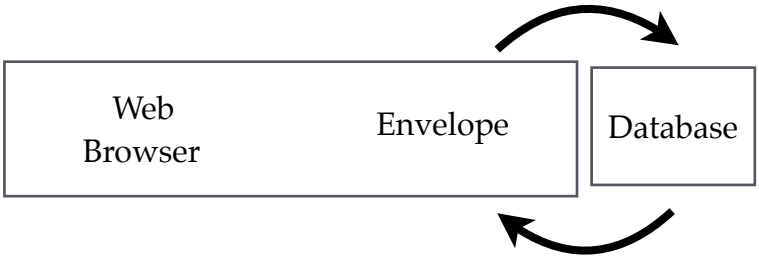
```
// accept request

// do we have permission?

// talk to database

// format and return data
```

Introducing "Envelope"
The first truly new web platform since the beginning of the web.

Envelope is a completely new platform that prevents the developer from writing code in the middleware by replacing it. This forces all code to be written for the web browser or the database. The Envelope architecture looks like the Middleware architecture.



However, in practice, a developer can build applications using the desktop architecture model. This greatly simplifies development resulting in near desktop speed. Additionally, there are numerous security, maintenance and performance improvements that are automatically available.




```
<script>
//Link to Greyspots Framework
</script>
<body>
  <gs-combo src="db.table"
     column="project_name">
     </gs-combo>
</body>

// No code anywhere else
```

Envelope Paradigm

In order to work like a desktop architecture, Envelope ships with a complete Javascript toolkit for building database applications. This Javascript toolkit, called "Greyspots", is based on a technology called "Web Components".

Greyspots is very different from other frameworks in that it provides HTML tags that can be used to display rich interface objects on a web page and connect those objects to the database without the developer having to write Javascript or Middleware code.

Introducing "Envelope" (continued)

The Greyspots web components are highly integrated with Envelope and PostgreSQL. This reduces development time down to desktop application levels. Additionally, much work has been done to the components to incorporate desktop-like application richness and mobile compatibility.

Unlike many frameworks that came before it, web component technology is capable of parity with desktop applications. The goal of the Greyspots framework is to reach absolute parity but a great deal of work has been completed and the goal is nearly accomplished.

For the first time a small web application can meet a small desktop application budget. This is accomplished by reducing total lines of code. When migrating an existing website to Envelope, the code savings easily exceeds forty percent.

Due to the discrete nature of components in Web Component technology, anyone with a thorough understanding of Javascript can create new rich user experiences. And when testing new experiences, developers won't experience an exponential increase in effort as under previous Javascript frameworks.

Envelope has additional advantages that exceed the scope of this document. Visit the Envelope website at www.envelope.xyz for more information. Envelope is fully documented, free and open source. Paid support is available.

Conclusion

We appreciate your attention. The information provided in this document was the very best we could provide as of the latest revision date, October 14, 2015.

If you have any feedback about this white paper please forward it to justin@wfprod.com or call 817-503-9545. We especially appreciate critical feedback because we want this document to be the best resource available concerning PostgreSQL-backed applications.

If you'd like to obtain more specific information about your situation please contact Workflow Products, LLC. We appreciate your consideration and will work hard to earn your business.

Best regards,

Justin Tocci
Programmer
Workflow Products, LLC
7813 Harwood Road
North Richland Hills, TX 76180
phone 817-503-9545