SpreadsheetWEB

Architectural Overview

August 2019

Pagos, Inc.
47 Third Street
Cambridge, MA 02141, USA
www.pagos.com

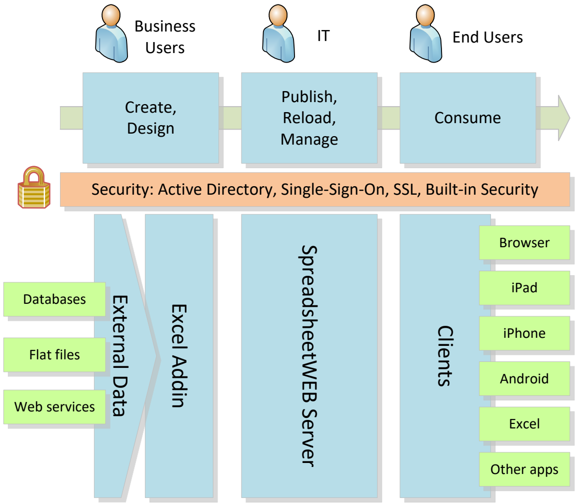
The information contained in this document represents the current view of Pagos, Inc. on the issues discussed as of the date of publication. Because Pagos must respond to changing market conditions, it should not be interpreted as a commitment on the part of Pagos and Pagos cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. PAGOS MAKES NO WARRANTIES, EXPRESSED OR IMPLIED, IN THIS SUMMARY.

Introduction

This whitepaper provides an architectural overview of the SpreadsheetWEB software.

SpreadsheetWEB is a platform that enables the conversion of Excel spreadsheets into web-based applications. While SpreadsheetWEB maintains simplicity in its integration with Excel, it can also be an extensive toolkit that aids in the creation of complex web-based applications, such as high-performance online calculators, web forms (which can be used to capture and analyze data with built-in support for dashboards and data-via-email), charting applications, and much more.



SpreadsheetWEB employs three primary methodologies for generating web applications: converted applications, web services, and Designer applications.

These definitions are outlined below:

- *Converted* applications are generated using an Excel add-in that mimics that user interface of the spreadsheet in Excel.
- *Web services* expose your calculation model via RESTful web service to be consumed by external applications.
- *Designer* applications utilize the Excel model as a calculation service but leave the end user to define their own responsive web front-end using the Designer module.

Converted Applications

For the *converted* approach, there are three primary components that define the process. It begins with the business users creating and designing their applications in Excel. Most business users are already familiar with the Excel environment and can easily develop tools using Excel. Using the SpreadsheetWEB Excel addin, business users design their applications. They can also integrate external data with their application designs.

Next, the final design is turned over to the IT department, who are typically responsible for publishing the application to the SpreadsheetWEB server and handling any future updates to the application. This department will also have access to SpreadsheetWEB's built-in Control Panel, which will provide all of the tools necessary for the management of these applications.

Once the application has been published to the SpreadsheetWEB server, the targeted end users will be able to access it through any supported client interface, including a web browser.

Web Services

For the *web service* approach, business users build their calculation model in Excel. Using Excel's built-in *named ranges* concept, users can define cell ranges of varying sizes as *names*. These can then be referred to as input and output from the web service and will constitute the basic I/O mapping when calling the RESTful web service for calculation results.

A key and set of credentials can be generated for a given web service application. These are then utilized to access the application securely with an access token.

The web service application resides on the SpreadsheetWEB web server, noted below. This application utilizes the same core engine for performing the calculations, which provides the same benefits of load-balancing – internal (nodes) and external (multi-server). Web service applications can perform statefully or statelessly, which means they can be configured to retain previously-set information or return the state back to the original between requests.

For web service applications, the Excel add-in can be utilized to define the *saving cells*, which can act as a means of storing and subsequently retrieving data through various web service methods; however, it is not utilized for any user-interface-oriented tasks, since the Excel model functions as a calculation service, rather than as a client-facing application in-and-of itself.

SpreadsheetWEB Designer

The Designer module is a web-based tool that can be used to fully design and customize responsive web frontends around an Excel-based calculation model. Similar to web services, the user defines their model in Excel and clearly designates a series of *named ranges* that will be utilized throughout the design process (e.g. binding inputs, print processes, email contents).

The model is then uploaded to the system as a new Designer web application and all subsequent configuration occurs through the Designer frontend.

Interactions with the module are transaction-based with a clear delineation between the *work-in-progress* (WIP) transaction and the *published* transaction. Published applications are accessible to the target client scope and are effectively immutable, whereas work-in-progress transactions are used to develop and maintain the application actively when changes are required. The process of publishing an application marks that transaction as finalized and releases the new interface and features to the target audience. This also ensures clear auditability between transactions and highlights the changes that were made as the application was developed and maintained.

The module is a separate web application from the Control Panel interface and uses Single Sign On (SSO) to facilitate authentication and various interactions with the system. It typically resides on the same web server as the Control Panel and the API.

The Designer module maintains its own separate database and each application that is created with the Designer can have its own database to store captured data. These databases typically reside on the same database server as the SpreadsheetWEB central database.

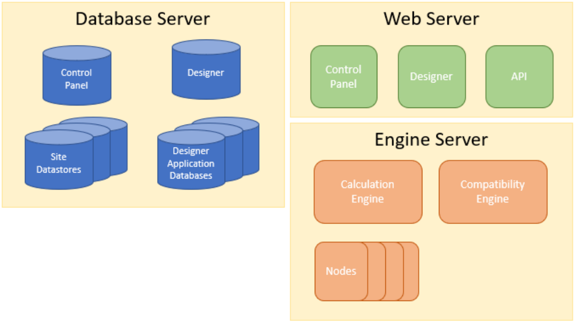
Server Software

The SpreadsheetWEB server software consists of four separate components, which communicate with one another and back to the database server. These are as follows:

1. The Control Panel is a web application that is utilized for user and application management. It is also responsible for generating the interfaces associated with *converted* applications.
2. The API (Web Service) is a web application that is utilized for calling Excel models as calculation services. It also handles authentication workflows and communicates directly with the Engine to facilitate calculation processing.
3. The Designer is a web application that is utilized for designing responsive and customizable frontends. It utilizes the API to facilitate calculations (and thus, indirectly, the Engine).
4. The Engine is a Windows Service that handles all of the calculations within the spreadsheet applications. It has built-in mechanisms for internal load balancing and state handling.

These components are designed for maximum scalability in order to host as much concurrent data usage as possible, granted any hardware constraints on these nodes (i.e. engine servers). They can either be installed on the same server or individual servers within the same network. The SpreadsheetWEB solution works natively on both x86 and x64 CPU architectures and benefits from the processing power of either system architecture.

The SpreadsheetWEB Control Panel and Engine software requires .NET Framework 4.5.2+ to be installed on the target server(s). Additionally, the Designer module requires .NET Framework Targeting Pack 4.6.

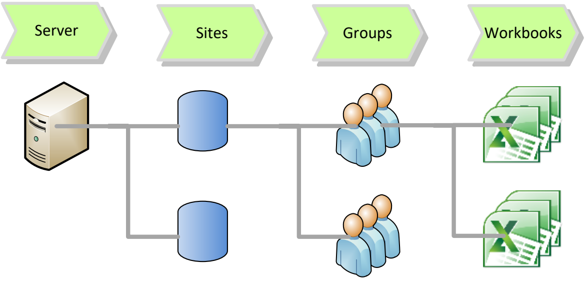


Web Server - Control Panel

The Control Panel constitutes the visible portion of the SpreadsheetWEB package, which is dedicated to the management of web applications. Although our server-side tools can be installed onto an internal network, this is generally the only part of the solution that maintains open access to the World Wide Web over Microsoft IIS.

Due to its naturally IIS-friendly design, it works with all well-known and mature security solutions without any additional software configuration or modification necessary . You may benefit from the use of a firewall, HTTPS, and other security protocols.

The Control Panel is accessed through user accounts, which are defined by the System Admin. In order to ensure the security of your data, all logistical information, including access logs, is organized and stored on an encrypted database table (in other words, we protect the usernames and passwords from any undesired access to the server's database). You may also benefit from access to a list of Active Directory users who are already defined on your network.



Each SpreadsheetWEB server installation forms a particular hierarchy that can be categorized into three levels: Sites, Groups, and Workbooks.

Engine Server

The SpreadsheetWEB solution converts your spreadsheet files in order make them functional on a web browser, with any special design mechanisms and instructions predefined as spreadsheet formulas. It employs a built-in calculations engine, developed in C++, in order to maximize performance and memory management.

The Engine software is a Windows service that controls the start-up and shut-down of its own application nodes for high-scaled spreadsheet calculations. The Engine service receives the requests from the Control Panel, performs the necessary calculations on the core spreadsheet node process, and then returns the results back to the Control Panel for rendering.

When a user opens an application, the web server sends a request to the engine server. If the workbook is already loaded onto the engine server, then the request is processed and the results are sent back to Control Panel and presented to the user. If the workbook has not yet been loaded, then the engine server requests the workbook from the database and loads it into the memory bank. The loaded workbooks stay in memory until no user activity is detected for a defined period of time. This time-out parameter can be set in the configuration files.

Architecturally, SpreadsheetWEB is built as a stateless engine. In other words, the same workbook instance on memory can handle requests from a multitude of concurrent users. While a single instance can handle multiple users, maintaining multiple instances of a workbook will produce better response times. As a result, the engine server can also be configured to have multiple workbook instances, even allowing each user session to be associated with an individual workbook instance. This approach will produce the best performance if each user session is running successive operations.

ObjectServer.exe is the process that handles application and workbook management on the engine server. The actual workbook calculations are handled by a separate process: SSWebNode.exe. There can be multiple SSWebNode.exe instances configured per engine server. Each SSWebNode can handle multiple workbooks for different applications, but can only maintain a single workbook for a given application. The ObjectServer can scale the number of SSWebNodes dynamically, depending on the available server memory and settings defined in the sswebnode.exe.config file.

ObjectServerGuardService.exe runs on the engine server and listens to the operations of the ObjectServer. If the ObjectServer fails to respond to the queries sent by the Control Panel, it can detect the situation and restart the ObjectServer. As a result, the system can recover quickly. User sessions don’t experience any data loss, since the session information is maintained by the Control Panel. If there are multiple engine servers in the system, any unresponsive user sessions are directed to another engine server while the failed server recovers. If there is only one engine server, the user sessions become unresponsive until ObjectServer fully restarts. In either scenario, user sessions will never experience any data loss. SpreadsheetWeb also supports SQLServer as session storage, so it is also possible to maintain user sessions even after a restart of the Control Panel.

Instruction transfers from the Control Panel to the Engine service are handled by WCF (Windows Communication Foundation) and are treated as secure data transfers. The instructions defined by the WCF service require unique application keys, which are only available within the database tables. This simple approach makes unsecured access to these instructions practically impossible.

WCF is also responsible for any communication between sub nodes and engine services, meaning that the aforementioned security points are also available, in addition to specific process targeting by defined ports (configurable per node) and the Windows process tracking handled by each process.

Database Server

The SpreadsheetWEB server also requires a database installation. The database maintains all of the Control Panel parameters, as well as application records within the database. The Control Panel system-related parameters are stored in the *SpreadsheetWEB* database. An additional database (*PSWDataStore_X*) is also created, which stores Control Panel and application-specific data associated with a given site. As there can be multiple sites associated with a single SpreadsheetWEB installation, “X’ in the database name denotes the site’s ID. For example, the database names would be entitled PSWDataStore_1, PSWDataStore_2, and so on, for each site associated with the SpreadsheetWEB installation.

SpreadsheetWEB can be installed with a Microsoft SQL or PostgreSQL server database. A Microsoft SQL installation can use an existing SQL server (the minimum requirement is SQL 2005). SpreadsheetWEB can also run against the free SQL Express edition, but this approach is not recommended for applications with large user bases.

As an alternative, PostgreSQL is a free and enterprise-ready database solution with a simpler and faster installation (in comparison to Microsoft SQL), since it will not require any additional configuration. The database files are copied by the SpreadsheetWEB setup package.

Dashboard Component

SpreadsheetWEB also includes a data visualization and business intelligence module built into the Control Panel. Microsoft Excel is the most widely-used software for data analysis, reporting, and visualization worldwide. However, it suffers from severe data size limitations. While creating visualizations and reports for small data sets may be very simple in Excel, it becomes substantially more problematic as the data size grows. In order to address this precise limitation, there is a growing number of data visualization and business intelligence solutions on the market that offer more flexible interfaces.

The Dashboard is SpreadsheetWEB’s integrated data visualization and business intelligence model. It is able to analyze large amounts of data without requiring the intermediary step of using Excel. It connects directly to the data source and allows users to create their own reports using a browser-based, interactive interface.

Client Applications

Workbooks on a SpreadsheetWEB server can be consumed by various client applications.

Browsers

The SpreadsheetWEB browser interface allows workbooks to be edited and viewed across all modern browsers. The implementation is primarily built in JavaScript, so no plug-in installation is required to access the workbooks. Workbooks can also be accessed through touch-enabled tablet devices running iOS, Android, or Windows operating systems through mobile browsers, including Mobile Safari and the Android browser. Some older browsers (e.g. Internet Explorer 6 and below) are not supported, as their limited JavaScript support may cause instabilities with existing SpreadsheetWEB features.

Third Party Applications

SpreadsheetWEB also offers an Application Programming Interface (API) for programmers to develop applications that interact with workbooks through SpreadsheetWEB.

Security

Pagos recognizes the importance of data security and applies various measures to protect your data and applications. All access to workbooks is handled through HTTP or HTTPS, depending on the server configuration. The SpreadsheetWEB server software also supports industry-standard, 128-bit Secure Sockets Layer/Transport Layer Security (SSL/TLS) or higher encryption.

Authentication

SpreadsheetWEB offers a built-in user management and authentication model. The user management module is only available to Admin-level users. The module offers remote administration, user creation, user role definitions, and application access privileges. Encrypted username and password information is stored in the database. Users can access the Control Panel and restricted workbooks through a native login page. It is also possible to configure advanced password rules, including length, usage of characters, etc. The system collects data regarding login and logout activity for auditing purposes. It also logs unsuccessful login attempts and locks the account after 5 failed attempts. A timeout period can be configured to automatically log out inactive user sessions.

SpreadsheetWEB also supports delegated authentication through Active Directory (AD). System administrators can manage their users centrally by disabling users and enabling new ones depending on an employee’s status.

SpreadsheetWEB supports Single-Sign-On (SSO), allowing users to access workbooks from a computer that they have already authenticated with a corporate login. This provides a seamless integration with corporate systems, where workbook links can be shared through other applications and portals.

Authorization

SpreadsheetWEB controls access to workbooks through group, policy-based authorization. Each user can be associated with a user role and multiple groups. Each user role is granted or restricted access to certain functionalities, such as the ability to edit, delete, and upload workbooks. User groups controls which workbooks a user is able to interact with. For example, a user who is a member of marketing and finance groups may have their access limited to workbooks under those groups.

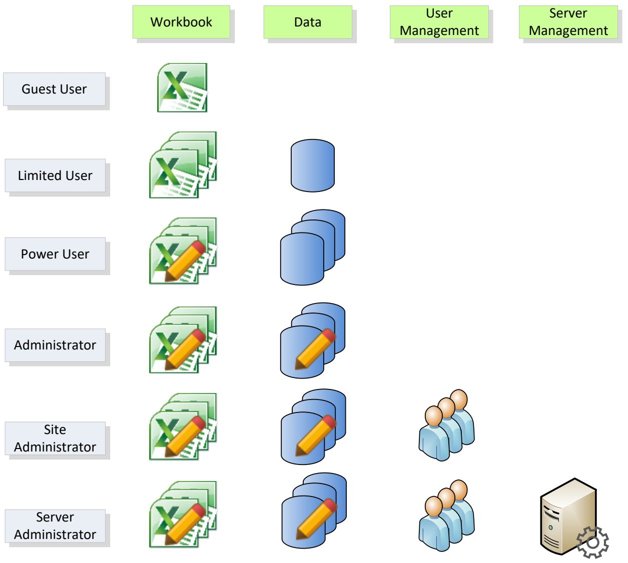
SpreadsheetWEB includes a set of built-in user roles. Additionally, the System Admin can create custom user roles by selecting permissions from a library of functionalilties.

Application Security

Each workbook on a SpreadsheetWEB server, whether associated with an application, web service, or dashboard, can be assigned a security level. This defines how users and user groups can interact with the given workbook. Access rights to workbooks (such as view, edit, upload, and delete) can be controlled. These permissions are managed through the Control Panel interface.

Data Security

SpreadsheetWEB handles workbooks and data separately. Conventional data handling, such as the method utilized by Excel, consists of a singular display for the user interface (the grid and the menu), business logic (formulas, cell connections, VBA) and data (embedded data in the grid). There is no distinction between these three components. On



the other hand, SpreadsheetWEB distinguishes and and handles these three components as separate entities. As a result, distinct security rules can be applied within each domain.

Data security has been a growing concern in the industry, resulting in part from a multitude of government regulations that require companies to secure and control access to certain types of data. Since this type of data (e.g. financial and personal information) has traditionaly been stored or intermittently copied into workbooks, meeting those regulations has become less and less plausible within the constraints of a traditional Excel framework. Since SpreadsheetWEB decouples the data from the workbooks, it allows companies to enforce stricter security measures for their data, even on the level of data cells or ranges.

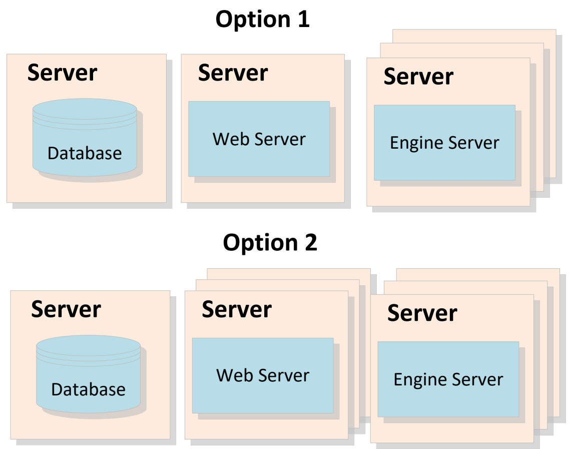
SpreadsheetWEB also keeps user logs, indicating when and how a user accesses a workbook. This allows companies to meet the criterion set by stricter auditing guidelines.

Scalability

SpreadsheetWEB supports configurations that contain multiple web and engine servers without requiring any additional load-balancing hardware. Mutiple web servers can be clustered using Microsoft Windows clustering. The engine server can be configured to support thousands of users through vertical and horizontal scaling.

Vertical scaling refers to the addition of hardware resources (i.e. CPU and RAM), in order to increase the Engine’s capability of handling larger user pools and more applications. SpreadsheetWEB can load the same workbook multiple times in server memory by creating multiple SSWebNode instances. Each instance can serve multiple user sessions by utilizing any available CPU and RAM on the server. Since each SSWebNode maintains user caches per workbook, having one workbook user per SSWebNode will lead to the best performance, especially with complex workbooks that require more input fields. For simple workbooks, the difference is negligible.

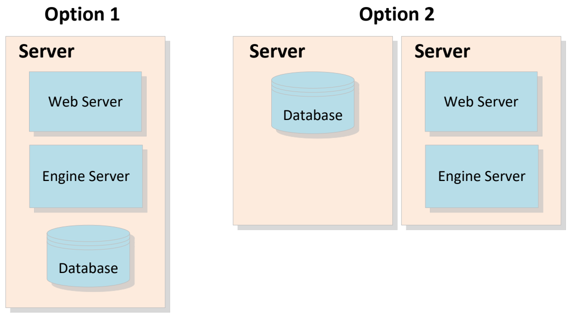
Horizontal scaling refers to adding more server resources, either virtually or physically, to a particular deployment. New servers can be added by simply updating the web.config file for the web server. The Control Panel can then utilize the added server capacity in order to distribute the user sessions.



Deployment Options

Single Server

All components of SpreadsheetWEB can be deployed on a single server. This is ideal for low-volume, low-usage applications. Alternatively, the deployment can be separated from the database server. Most organizations have database servers that are used for multiple purposes and SpreadsheetWEB can utilize an existing SQL server deployment. The installation packages of SpreadsheetWEB include the database scripts necessary to deploy on an existing SQL server.



Multi Server

In this configuration, web and engine servers are deployed on seperate machines to scale the calculations. The SpreadsheetWEB engine can be installed on multiple servers with each instance connected to the same web server. Requests to the spreadsheet engine nodes are distributed by SpreadsheetWEB. This alternative is well-suited for applications with moderate usage that still incorporate complex spreadsheet models that take longer to calculate.

SpreadsheetWEB servers can also be clustered using Microsoft Windows Server clustering options. The SpreadsheetWEB Engine can also be deployed on multiple nodes. This deployment is ideal for high volume and business critical applications. Clustering will improve availability, even if a web server becomes unresponsive.