AppSheet

AppSheet:
Design and Architecture

ABSTRACT

This white paper introduces the design and architecture of the AppSheet platform. Readers will gain an understanding of the AppSheet operating infrastructure, platform architecture, information architecture and security model.

Table of Contents

Executive Summary

NO CODE APP DEVELOPMENT

Mobile devices are the primary productivity platform for today's enterprise workers. Despite the explosion of consumer mobile apps, many business productivity apps are still stuck on the web or the desktop. This is because the cost of app development is too high, it takes too much time, and it requires specialized engineering skills. AppSheet's no code app development platform breaks through these barriers and enables anyone in an organization to build and customize apps for themselves, their teams, and their partners.

It is very important that business apps integrate with existing data sources within the enterprise whether departmental data sources like spreadsheets, cloud-based services like SalesForce, or IT data sources like SQL databases. Business apps need to be customized to fit the specific workflows and processes that are already in place in the organization. And finally, business apps need to be modern, elegant, and performant, matching the expectations of employees who use mobile apps on an everyday basis in all other aspects of their lives.

AppSheet is a powerful platform that ensures that apps meet these three crucial requirements.

This paper provides an overview of AppSheet's application model and platform architecture. It also describes how AppSheet conforms to well-established operating processes, especially in the areas of reliability, information security, and privacy.

ABOUT APPSHEET

AppSheet, a leading no code mobile app platform, offers an innovative data-driven approach to the creation, deployment and management of mobile apps. Any member of an organization can create mobile apps as easily as creating documents, presentations, spreadsheets, or web pages. Teams and organizations use AppSheet to drive productivity through custom mobile apps. A mobile app project no longer requires a large budget, a long schedule, scarce mobile developer resources, and complex planning. Instead, any team or business owner can build an app themselves without development skills or training. App development, testing and improvement is rapid and iterative with changes made in minutes and deployed instantly. A broad range of apps can be built to serve the needs of various business functions. The platform is integrated with leading cloud-based data sources such as Google Drive, Office 365, Box, Dropbox, Salesforce, SQL Server, PostgreSQL, and AWS DynamoDB.

AppSheet's proven track record has made it a compelling choice for more than 20,000 organizations around the world. Customers include small- and medium-sized businesses, Global 2000 companies, academic institutions, as well as local and federal government agencies.

AppSheet Concepts

The app creator goes through three steps to create an app.

Connect to your data

Customize your app

Deploy your app

1. **Connect to data:** pick a data source like a spreadsheet or SQL table. AppSheet automatically extracts the structure of the data and automatically generates a working app from the data.

2. **App customization:** configure and modify the structure of the data, the presentation elements, or the app behaviors. Each of these changes is code free and instantly reflected in the app.

3. **Live test deployment:** test the app on a mobile device or share with other users to test or use.

Typically, these three steps happen in the first few minutes of AppSheet use. Subsequently, the app creator repeatedly adds further data or configures the app, adding various features and capabilities to the app. In fact, even after the initial version of the app has been deployed to live users, subsequent improved versions of the app continue to be developed in this rapid iterative fashion.

How is it possible to instantly create an app and instantly deploy it? This requires an explanation of the underlying app architecture. There are two structural components to a working AppSheet app:

1. **The App Definition:** this metadata represents all the information provided by the app creator while defining the app. It includes the data sources, the configuration choices, and behavior settings specified in the process of app creation. The App Definition is maintained in the cloud-based AppSheet service.

2. **The App Data:** this is the actual data that the app displays, captures, and processes. The data resides in the backend data source of the organization (a cloud-based file system like Google Drive or Office 365 for a spreadsheet source, or a cloud-based database system like SQL Server for a database table).

Your Data

Cloud Storage

App definition

AppSheet Service

Local Storage

Your Device

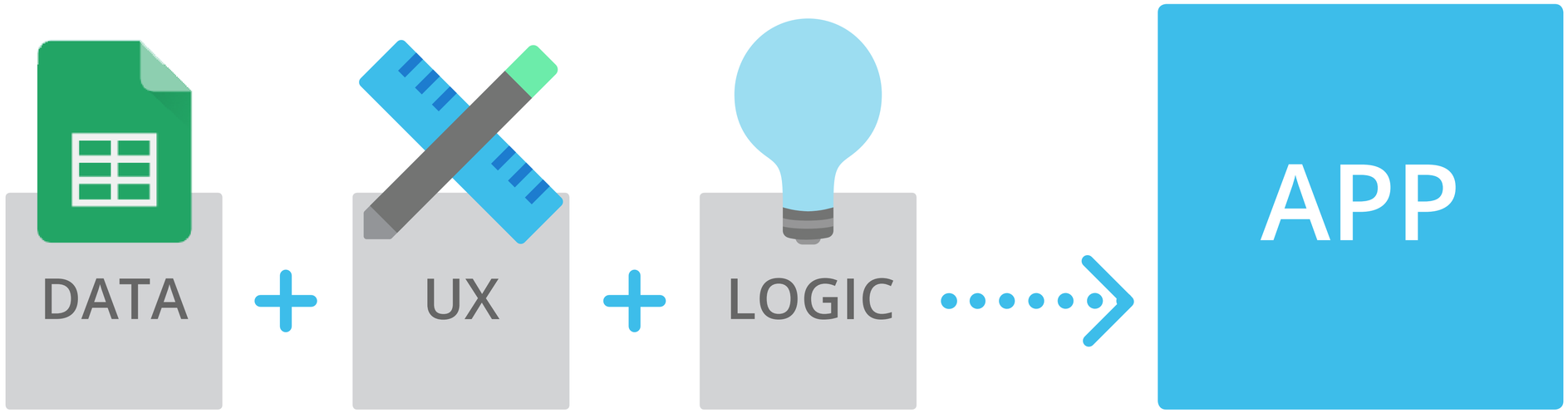
An app on a mobile device runs within a prebuilt AppSheet mobile app host. It communicates with the AppSheet cloud service to fetch the App Definition as well as the App Data (which is fetched from its original cloud storage location). The mobile device maintains local copies of the App Definition as well as the App Data to ensure that offline execution is possible. Any changes made to the data in the app or at the backend are synchronized when appropriate.

The AppSheet App Model

The AppSheet App Model represents the set of features and capabilities that can be expressed in an App Definition. Every App Definition is an instance of the App Model.

Customers often have a question: *"What sort of apps can I build with AppSheet?" or "Can I do X with AppSheet"?* These questions are really about the expressive power of the AppSheet App Model.

This white paper does not describe the details of the App Model. It is described in detail in our technical documentation and is evolving rapidly as we add new features and capabilities. For the purpose of this white paper, it is useful to be aware that the App Model (and every App Definition) has three components, each with a rich array of capabilities: (a) Data definition, (b) UX definition, (c) Behavior or Logic definition



APP DEFINITION SECURITY MODEL

The app creator must decide if the app is to be secured (accessible only to specific users) or open to all users. For most corporate internal apps, the former option is appropriate.

When an app is configured for secure access, there are two consequences:

1. The very first user interaction in the app is a sign-in screen, where the user is asked to authenticate using one of the standard authentication mechanisms supported by AppSheet.

2. Once authenticated, AppSheet also checks if the user is authorized to access the app by comparing the user's credentials with an explicit user whitelist maintained for every app.

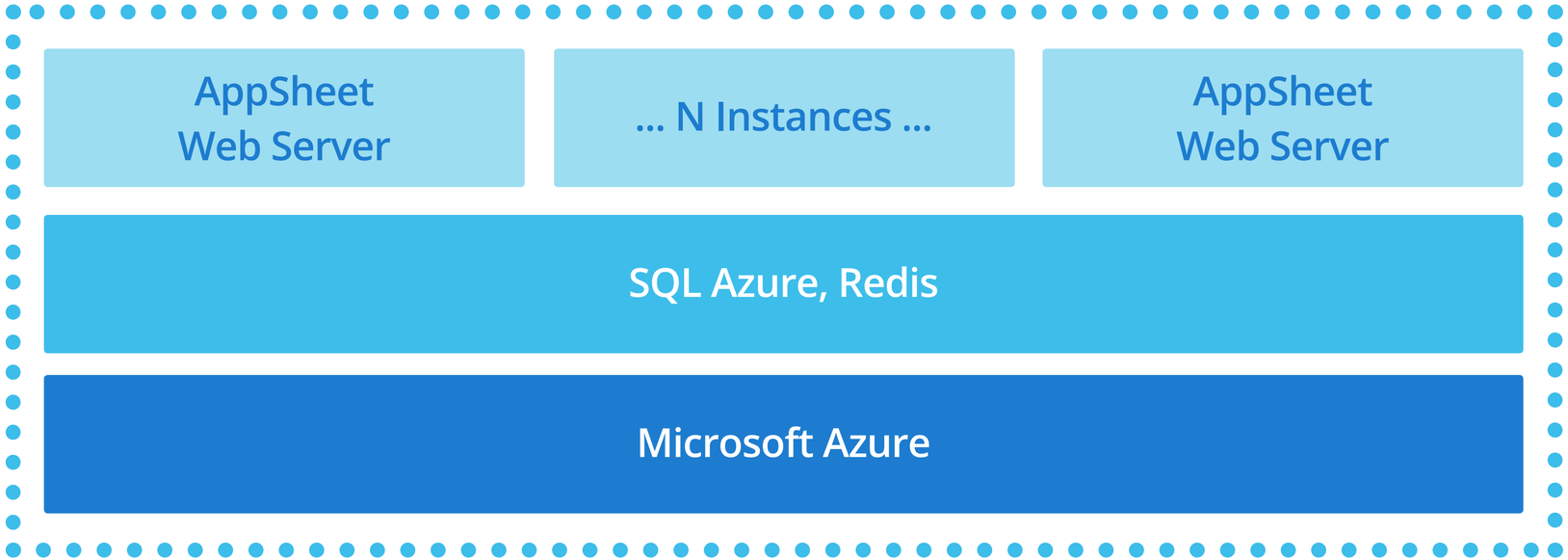
App usage is only possible if both the authentication as well as authorization succeed.

Platform Architecture

The AppSheet platform has two components the cloud service backend and the mobile device client. All communication between the components is encrypted and uses the HTTPS protocol.

CLOUD SERVICE BACKEND ARCHITECTURE

The AppSheet cloud service runs in a set of identical redundant web servers hosted in the Microsoft Azure cloud infrastructure.



Each web server exposes a secure REST API that is used by the mobile device client to communicate with the server. The servers store user profile information and app definitions in a shared SQL Azure instance. The servers also share a common main-memory Redis cache for performant access. All of the backend server infrastructure is hosted in the Microsoft Azure cloud utilizing its automatic data backup and server reliability infrastructure.

The AppSheet backend is not a persistent repository for App Data. The App Data resides in its original cloud-based data source. The AppSheet backend fetches the data when it is necessary to synchronize with the mobile clients. The data may be cached in the web servers or in Redis for short periods (of the order of minutes) for performant access. However, App Data is never persisted by the AppSheet backend.

The backend records an audit log of all data changes made by app users. This log is saved in Azure tables. If there is important “PII” content that should not be logged, the AppSheet model allows the app creator to indicate this, and the PII content is explicitly excluded from the audit log. As an additional safeguard, the lifetime of the audit log can be explicitly controlled by the app creator.

The AppSheet backend does not maintain its own user authentication i.e. it does not maintain a database of usernames and passwords. Instead, all user authentication utilizes external user authentication services like Google, Office365, Dropbox, Box, or Smartsheet, using the OAuth protocol. Once the user is authenticated using OAuth, their access tokens are persisted in the SQL Azure database. The AppSheet backend ensures that each app utilizes the appropriate OAuth access token when accessing the App Data from its cloud-based data source.

MOBILE DEVICE CLIENT ARCHITECTURE

The mobile device client is an interpreter that can execute any valid App Definition. It is built primarily with HTML5 and hosted within a native app wrapper on Android and iOS. This allows AppSheet apps to run on Android, iOS, and in any modern web browser (a few app features that are device specific will not work in a browseronly environment). Apps execute identically on iOS and Android, with the same user interface and the same behavior.

A generic version of the Native App Wrapper is already available in the iOS and Google Play app stores. It can run any App Definition and is used both for app testing and for lightweight deployment. Some customers prefer to create a standalone whitelabel app that can be independently deployed and managed in the app stores. AppSheet enables this option as well. The architecture of each whitelabel app is identical to the generic version, except that it is configured to run exactly one app.

When the user makes changes to the data via the app, the changes may be immediately sent to the backend. More commonly, the changes are made locally on the device and queued up to synchronize with the backend. This enables completely offline execution of the app, and also handles occasional loss of connectivity. Mobile devices cannot be guaranteed to have network connectivity. Consequently data synchronization could fail in unpredictable ways. AppSheet has designed synchronization protocols that ensure that (a) the same queued action can be synchronized multiple times without compromising correctness (a property called “idempotence” of actions), and (b) data changes made by the client are never lost due to poor connectivity.

Operational Processes and Policy Compliance

The AppSheet backend is hosted on Microsoft Azure's cloud infrastructure. Several issues of security and privacy compliance, especially related to hardware infrastructure are addressed by this document provided by Microsoft Azure:

https://www.microsoft.com/en-us/trustcenter/Compliance/default.aspx

DATA BACKUP

The data maintained in AppSheet's backend SQLServer database (user profiles, access tokens, app definitions, and usage statistics) is backed up automatically once every 24 hours. Older versions of app definitions are also redundantly maintained in Azure table storage and can be recovered and restored by the app creator.

The data backup architecture is therefore resilient not only to server failures but more importantly to user error.

SERVER RELIABILITY

AppSheet runs multiple redundant web server instances, ensuring that the availability of any one web server does not negatively affect the overall availability of the service. This has resulted in a highly reliable web service with 99.99% monthly uptime.

INFORMATION SECURITY

Customer service and customer support are not limited to a specific team within AppSheet instead, every employee takes turns engaging in customer support. This is a fundamental tenet of our company's value system and a way to emphasize the importance of customer satisfaction.

Part of customer support is troubleshooting issues faced by customers as they build apps. There are internal tools available for employees to examine the app definition of a customer, or the audit log history for a specific app. These internal tools allow an employee to run a customer's app for the purposes of debugging. Additionally, each employee may explicitly transition into an administrative mode that "impersonates" a specific customer in order to debug issues. This capability is used rarely and only for the purpose of troubleshooting a pending customer issue. A customer can request that their app (and the data in it) be placed on a exclusion list that makes it inaccessible to these internal troubleshooting tools.

Conclusion & Resources

The AppSheet “no code” mobile app platform unlocks the creativity, innovation, and productivity potential of all employees of an organization. Any employee or team can easily create mobile apps to make their work or their team more productive.

It is important for IT professionals to understand the design and architecture used to make this happen, especially since AppSheet apps are being used with business data o solve business problems. This paper describes the various elements of design and architecture that form the basis of the AppSheet platform. In particular, the paper clarifies issues of access control, privacy, information security, and reliability.

For further information, contact AppSheet Sales at **sales@appsheet.com.**