Creating a user-centric mobile hub with Xamarin and Azure App Service

Microsoft has more than 114,000 employees who work at offices and campuses all over the world. To improve the employee experience, increase work satisfaction, and drive productivity, Microsoft campuses feature enhanced amenities and service offerings. Employees have access to services such as shuttles, shopping kiosks, athletic facilities, and full dining experiences with online ordering. For new employees, it can be a little overwhelming to fully grasp what services are available to them. And long-time employees may not be familiar with, or think about using, new services as they become available. Information about all the services is available on company intranet sites, but booking and ordering information was somewhat dispersed across the different service sites.

At Microsoft IT, we created EmpEx, a mobile app that brings together various employee experiences, such as booking a shuttle to travel to various office buildings, ordering online from the nearest building café, and even viewing a summary of their benefits information. We built EmpEx on Microsoft Azure platform as a service (PaaS), using Microsoft Visual Studio Team Services and Xamarin. And we designed it to interact with employee services on Windows 10 Mobile, iOS, and Android devices. The app integrates with existing IT service data feeds and Azure services like authentication, notifications, storage, APIs, and analytics. We used HockeyApp to distribute EmpEx and its updates to employees during testing. In addition to streamlining our app publishing processes, HockeyApp also collects useful telemetry and app performance information.

Different campuses offer different services, so we wanted the app to customize the user experience based on each employee's profile and current location to help them access available services, whether they're in their own workspace, at another office or campus, or even in another city.

When we developed EmpEx, our goals included:

- Providing snack-size experiences for everyday things employees care about.
- Empowering employees to work from anywhere.
- Orienting employee experiences to their context.
- Carrying interactions across places, time, and people.

Using Xamarin for cross-platform mobile app development

We envisioned EmpEx as a modern, intelligent app. To develop it, we needed to use an extensible platform that enabled our developers to create a comprehensive solution for Windows 10 Mobile, iOS, and Android devices without building and managing three distinct apps. Xamarin provided an entirely new ecosystem of platforms for building cross-platform mobile apps—it offers a single language, C#; a class library; and a runtime that works across all three mobile platforms.

We were interested in using a cross-platform native application approach so employees could easily use the app on their smartphones from any location. Using Xamarin, we can quickly respond to organizational requests by quickly delivering EmpEx and its subsequent feature updates much faster than would've been possible using traditional application development methods.

Single code library

Using Xamarin, we created and maintained a single code base for the back-end of the solution. The Portable Class Library layer, contains all the business logic and the code library, and it enables code sharing across Windows 10 Mobile, iOS, and Android platforms.

We can write once and run anywhere; thereby developing apps with speed and agility that's integrated with the supporting Microsoft Azure cloud platform.

Platform-specific design guidelines

We used Xamarin and Visual Studio to develop the rendering based on platform-specific user interfaces (UIs). We created a set of platform-specific design guidelines, based on the best practices in publicly available SDKs for Windows 10 Mobile, iOS, Android, and our own policies. For example, the devices have different sign-in methods; the publicly available SDK includes guidance for enabling biometric, or fingerprint, sign in, but we needed to consult our own polices to develop specific guidance about the other conditions that must exist for us to use it as an application authentication method.

Streamlining deployment using HockeyApp

We use HockeyApp to easily publish new builds of EmpEx and make it available for employees to install. HockeyApp integrates with Visual Studio Team Services (VSTS) and serves as more than just a repository for our application builds. It also distributes the app to employees and helps in logging telemetry, crash reports, and deployment release (flighting) information.

When we need to release a new build, the developer makes the changes in Visual Studio and submits the changes via a pull request on VSTS for code review. After the code changes are approved, they're checked in and the auto-build is kicked off. After the package is signed, it automatically gets published to HockeyApp. This saves us from creating download packages and publishing them to each app store. We also don't have to build a notification pipeline because HockeyApp notifies the employees who have installed the app.

Employee experience app architecture

At a high level, the EmpEx app takes information from existing service data feeds, as REST-based APIs. It renders the data feeds into device-specific UI layouts using *cards* for each service. After an employee authenticates using multifactor authentication, the different cards loaded with the appropriate data are displayed from each available service. Figure 1 shows the high-level architecture of EmpEx.

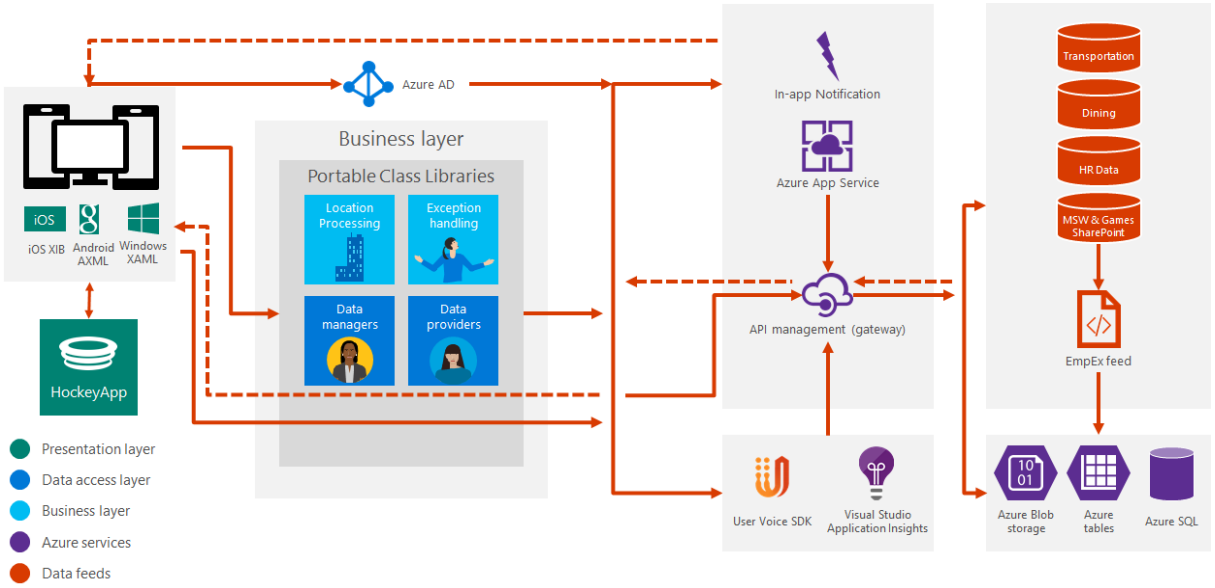

Figure 1. High-level architecture of EmpEx mobile app

Presentation layer

Within the presentation layer, we used the following components to render the UI for each platform:

- **Windows Phone XAML.** This renders the UI for Windows 10 Mobile devices.
- **iOS XIB.** This renders the UI for iOS devices.
- **Android AXML.** This renders the UI for Android devices.

Business layer

The business layer contains all the business logic as Portable Class Libraries. It's used by the EmpEx app façade and includes:

- **Location processing.** Location awareness helps to make the experience relevant to employees. We use a dual approach for location awareness. First, EmpEx looks at employees' region and location information within our HR system. For services that require physical location, like shuttle and dining services, it also uses GPS lookup.
- **Exception handling.** HockeyApp and the Application Insights SDK collects and logs exceptions to Application Insights that help us increase the overall reliability of EmpEx.
- **Cross-platform operations.** Consumes the models we created for view mapping. User settings and configurations are stored in Azure Table storage; they uses geo-location and employee role information to map the different EmpEx service cards to display.

Data access layer

EmpEx implements the Model-View-View Model (MVV) design pattern to provide a clean separation between user interface controls and logic.

In the data access layer, we used multi-factor authentication in Azure Active Directory (AAD) for authentication to the app. AAD validates the user credential and modules, and then it provides a token when authentication succeeds. The app also gathers telemetry and sends it to the HockeyApp and Application Insights.

The data manager calls the data provider module to retrieve models and create the view models and bindings needed to load the cards.

The data provider uses the validated authentication token to get the requested feed data and creates the appropriate view models. The view model contains the data models and validation logic from the respective data feed services.

Data feeds

For every service card in EmpEx, the app relies on a data feed from an internal, Azure-based, employee service. Employees can access these services through the EmpEx app:

- Time reporting information about vacation availability and accruals.
- Human resources, including benefits overviews, investments and savings information and holiday calendars.
- Transportation services, including campus shuttle and commute connectors.
- Dining services, including café lists, menus, food trucks, and so on.
- Facility services—office and building information, furniture options, heating and air conditioning and other general facilities information. Employee profile, including employee-specific internal profile, organization chart, and professional connections in a graph format.
- Activities and games.
- SharePoint services that deliver stories from the internal company news site.
- MSApprovals service that's used for action notification for purchase order approvals, reimbursements, and other requests.

The EmpEx feed acts as gateway, in addition to sending the multifactor authentication token. It stores and feeds user preference configuration and settings information to the app.

Azure services

EmpEx uses the following Azure services for storage and APIs:

- **Azure Storage.** This stores the NoSQL card metadata settings and configuration information. The app uses Azure Tables storage and Azure Blob storage:
  - **Azure Table storage.** We used Azure Table storage, a NoSQL key-value store, to store each employee's user settings and configuration, geo-location, company code, role, and service card mapping.
  - **Azure Blob storage.** Images that are used in the app are kept in Azure Blob storage.
- **Azure SQL.** We use Azure SQL to extract the key telemetry events from Application Insights when we want to look at trends for time periods longer than 30 days.
- **Azure Active Directory.** The app uses Active Directory Authentication Library (ADAL) to communicate with AAD to authenticate users and securely call web APIs by using OAuth 2.0 across all platforms.
- **Azure App APIs**. We use RESTful Azure App APIs for underlying data services. RESTful APIs provide resource representations that are easy to programmatically consume.
- **Application Insights.** We use Application Insights to gain insights through app performance telemetry and analytics.

Leveraging Azure service offerings reduced the amount of effort that would have been required to configure and maintain multiple on-premises servers. In addition, Azure gave us the platform to build the app that can support multiple operating systems, integrate with cloud-based services that are managed and monitored using Azure API management, and the flexibility to scale as more employees use the app.

Managing on-going development and release cycles

We're continuously integrating new features and services into the app—as well as regularly updating it—on all three platforms. For each development cycle, or sprint, we create a plan, develop the code, create telemetry events for new

features, and test the new build. After sign-off, HockeyApp notifies employees who have EmpEx installed that a new version is available.

During the planning phase, we look at our backlog. It includes information from telemetry and logging in Application Insights about the top app crashes, new card requests, and other feature enhancements in VSTS. The product owner and development manager prioritize what features and fixes we should include in the next development cycle. The prioritization is based on the telemetry data and feedback that's gathered in the User Voice functionality within the EmpEx app. We prioritize features and prepare the next development sprint's backlog, and the team creates an estimate and cost breakdown of the work. Based on the team's capacity, we include some of the requested items in the current sprint. We break down the features into small tasks—and those tasks are what drives the development changes.

The small tasks identified during sprint planning are developed into the code, and then they're tested. As we build new features into EmpEx, we also create new telemetry events to collect usage information about the new features. Telemetry data helps us understand how employees are navigating through the EmpEx features. It also gives us an idea about areas in which we can improve the experience or add new features.

We track diagnostics and debugging information using Visual Studio. The developers test the app's overall functionality with test scripts on release candidate builds before it's released to beta users—they're early adopters who run a pre-release version of the app to further validate functionality and provide feedback.

We use Power BI for analysis of user trends. It provides a dashboard view of the data from the Azure SQL tables on app usage, users enrolled, and daily and monthly active users.

Personalized employee experience

Location information is combined with role information to deliver a customized EmpEx experience for each employee. The app also has features that help employees further personalize their experience. Employees can sort the order of the displayed service cards based on their preferences. For example, if they take the shuttle often, they might want those cards displayed prominently; and if they frequently use EmpEx to access dining services information, they might want those cards displayed most prominently. They can even turn off cards for services they don't use.

We also added a feature that helps employees protect their personal information and data. For personal, organizational, and benefits summary information, the views that include actual sensitive data are hidden until the employee wants to display them on their screen.

Enabling user engagement

We have two ways that we enable user engagement, Azure notification services and HockeyApp app notifications. We use Azure notification services to send notifications from the app itself. Notifications are used to keep employees informed about statuses of their requests—such as where their shuttle is or the status for finishing a facilities request ticket. And we use HockeyApp app notifications when there's a new version to install.

EmpEx integrates with User Voice APIs so that employees can give us feedback and tell us what they'd like to see included in the app. We take the most requested items and add them to the VSTS feature request backlog. When employees send in an idea or vote on a proposed item, we communicate back to them to let them know that their suggestion is being considered or is being included in a future release.

Conclusion

Using Xamarin, it took roughly five months for three developers to build, test, and release the first iteration of EmpEx. This represented an almost 60 percent savings in developer efforts over having to create three separate apps.

HockeyApp, Visual Studio, and Application Insights integration with the HockeyApp SDK gives us more meaningful crash reports and stack traces that help quickly investigate issues while testing. Users also always have the latest

version and get automatic app updates, while automatic bug creation enables faster investigation. Because the app is integrated with Application Insights integration, we have rich monitoring of service performance and availability.

Using Azure APIs, Azure Storage, and Azure services enables mobile backend as a service. The shared backend features, like notifications and APIs that connect to diverse data feeds and the storage services, have easily and consistently helped create the cross-platform mobile app.

With Visual Studio Team Services, we've integrated source code management, and we create cross-platform builds for continuous integration and app distribution with HockeyApp, which also facilitates feedback and learning.

Historically, creating a cross-platform app would have required designing, developing, and deploying three different apps, all with their own code base that would have to be managed separately. For an employee services mobile app, the return on investment is primarily realized indirectly, through increased productivity and employee satisfaction. Without the substantial time, effort, and cost savings made possible through our use of Xamarin, Visual Studio, and HockeyApp, it would have been difficult to justify offering a cross-platform mobile app.

For more information

Microsoft IT

microsoft.com/itshowcase

© 2017 Microsoft Corporation. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.