Building a Banking App in the Cloud Made Simple with Azure

Introduction to Open Banking

Daniel Krzyczkowski

Principal Software Engineer

7 min read • Jul 27, 2020

Nowadays, many people have several accounts in different banks. For each bank, we have an application (web or mobile) where we can sign in and perform different operations. We can, for instance, make a bank transfer or access transaction history. And there are many functionalities beyond that.

When managing our budget and controlling spending, having transaction history in multiple places can make it difficult to track our expenses. Having a single place with all the data can be helpful. This is where the **PSD2 directive** can help. It allows us to develop applications that we can use to manage all our finances at once.

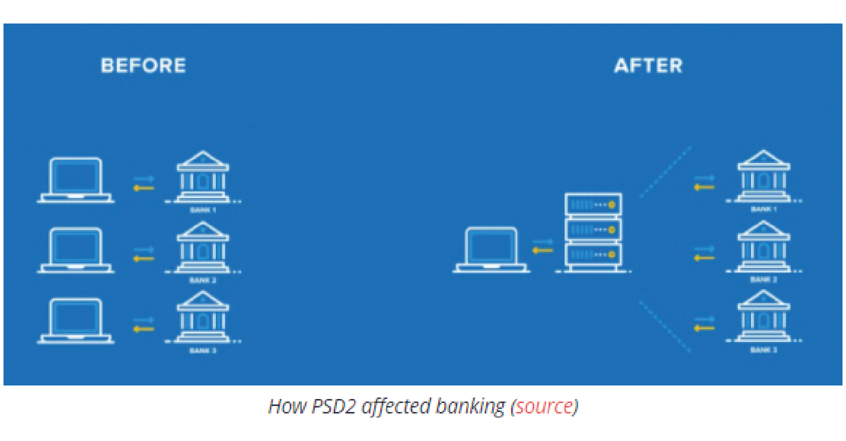
Contents:

Key points:

- What are the biggest challenges with mobile banking application development?
- What is the PSD2 directive?
- Which architecture style should be considered when building a banking application in the Azure cloud?

What is the PSD2 directive?

PSD2, or the second Payment Services Directive, was introduced at the start of 2018 by the European Parliament. It **forced banks to share data** between themselves and (approved) third parties, so new challengers and disruptors could enter the market more easily.



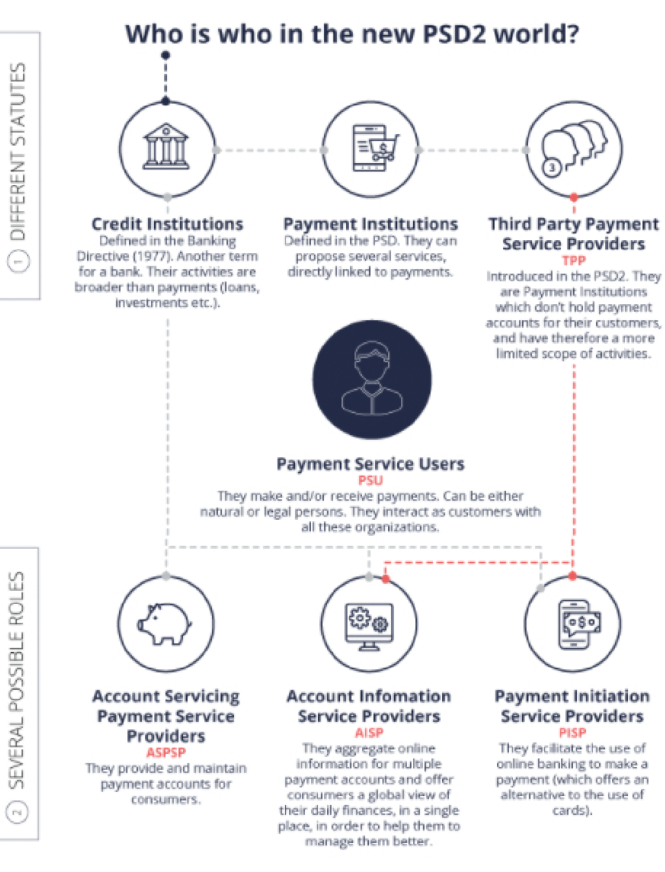
How PSD2 affected banking (source)

PSD2 allows banks to:

- securely share customer account and payment data
- with customer consent
- with regulated third parties
- through APIs.

As we can see, the PSD2 directive makes it possible to **aggregate data from different bank accounts.** When talking about such aggregation, we have to mention Account Information Service Providers (AISP). They can securely collect data from multiple accounts.

See the diagram below for more information about the **roles** in PSD2:



Various roles according to the PSD2 directive (click to view full-size)

Why choose Azure for financial services markets?

When talking about a solution that's handling banking data, the first thing that comes to the mind is **security**. Some core issues should be taken into consideration when implementing an app of this type. They include:

- **Authorization** – authenticated users can only access the business functionalities which they are allowed to
- **Data confidentiality** – the application should store sensitive data in a dedicated and secured file system. Sensitive information cannot be leaked through logs or error messages
- **Authentication** – there should be a strong authentication mechanism to verify user identity. Multi-step authentication (MFA) is recommended
- **Encrypted connections** – all connections initiated by the app should be encrypted for safety. HTTPS protocol should be used to connect to the cloud
- **Encrypted assets** – all the important data files, like property file or configuration file, should be hidden and encrypted.

There are also some important challenges, such as:

- **High availability and scalability** – the solution has to be available 24/7
- **Automatic user account and data removal** – when requested, user account and all their collected data should be removed
- **Modern notifications** – users on mobile devices should receive targeted push notifications
- **Data security** – all user data is encrypted at rest and in transit.

Using Microsoft Azure to build your application allows you to solve these issues. The cloud provides all the services you need to address these challenges.

Time for a real-life scenario!

Together with my team at Predica, we are working on a **modern banking application development** project for one of the biggest banks in Europe. It is a huge undertaking, requiring aggregation of data from different bank accounts into a single application.

The goal is to develop web and mobile applications which will enable users to sign in and see transaction history from different, connected bank accounts, so they can **plan their budgets** for the upcoming months.

A mock-up of our banking application interface

The building blocks of an Azure-based banking application

Let's discuss the role the **Azure cloud** plays in building a modern banking application.

Microsoft Azure provides many useful services that make the solution work smoothly and effectively. Here are the examples of services that can be used to build a modern, **high-availability banking solution** in the Azure cloud.

AZURE CONTAINER REGISTRY

Docker is becoming more popular nowadays. Containerized applications can be easily shipped with all dependencies. Azure Container Registry stores and manages private **Docker container** images. It provides easy integration with Azure Kubernetes Service described below.

AZURE KUBERNETES SERVICE

Kubernetes is a service that is responsible for automating deployment, scaling, and management of containerized applications in the cloud. It can be helpful when we're using a style of architecture based on **microservices,** like we are in this project. In Azure, Kubernetes is available as Azure Kubernetes Service (AKS).

It helps to build resilient and highly available microservice solutions in the Azure cloud. Back end microservices are located there.

It is worth noting that if an application running in a Docker container fails, Kubernetes will replace it and start a new instance automatically.

AZURE SERVICE BUS

Azure Service Bus is most commonly used to decouple applications and services from each other, and is a **reliable and secure platform for asynchronous data** and state transfer. It can be used to exchange information between different services. Data is transferred between multiple applications and services in a binary format, in a file that can contain JSON, XML, or just text.

AZURE API MANAGEMENT

This is a cloud gateway to communicate with internal micro-services (APIs). With Azure API Management it is possible to set up **inbound and outbound policies,** so we can cache response data or verify authorization tokens.

AZURE APPLICATION INSIGHTS

Complex solutions require good monitoring. With Azure Application Insights, we can **analyze and detect anomalies** in our solution, so we can quickly react and fix any issues.

AZURE WEB APP

Azure Web App is a dedicated service for hosting web applications, REST APIs, and mobile back ends – **first-class support** for ASP.NET, ASP.NET Core, Java or, Angular. It works on a global scale with high availability – scaling up or out manually or automatically.

AZURE NOTIFICATION HUB

Push notifications are a part of every modern mobile application. Azure Notification Hub enables **sending notifications to any platform** like iOS, Android or Windows. We can leverage Azure Notification Hub to send notifications related to updates on budgets.

AZURE ACTIVE DIRECTORY B2C

User authentication and authorization are crucial and access to the solution should be available only for a verified user. Managing user accounts, resetting passwords or editing profiles can be challenging. This is where Azure Active Directory B2C can help and simplify the process.

AZURE SQL DATABASE

An application handling sensitive data requires **secure storage.** Azure SQL relational database is the right choice here. **Advanced Threat Protection** can be added to discover and classify sensitive data, manage database vulnerabilities, and detect anomalous activities. Azure SQL enables data replication so we can build high-availability, resilient solutions with it.

AZURE STORAGE ACCOUNT

In some cases, users should be able to upload files. This information should also be kept safe. The Azure Storage Account service can help us with it. All data sent to Azure Storage is **encrypted** by the service.

AZURE FUNCTIONS APPS

Azure Functions are very useful in creating **automatic triggers** for specific tasks like calling another Azure service when a file in an Azure Storage Account is deleted. It can also be a good choice for integrating with Azure Active Directory B2C to provide additional user attributes from the external store during the login process.

AZURE KEY VAULT

Azure Key Vault is secure storage in the cloud where **sensitive credentials** are kept.

How are we combining these elements?

To build the solution we're using the Azure cloud services I described above. First, we did a **deep analysis of all functionalities** that should be included to plan user interface in both web and mobile applications. We defined and implemented many (more than 13) different microservices, including:

- Budget microservice
- User microservice
- Help microservice.

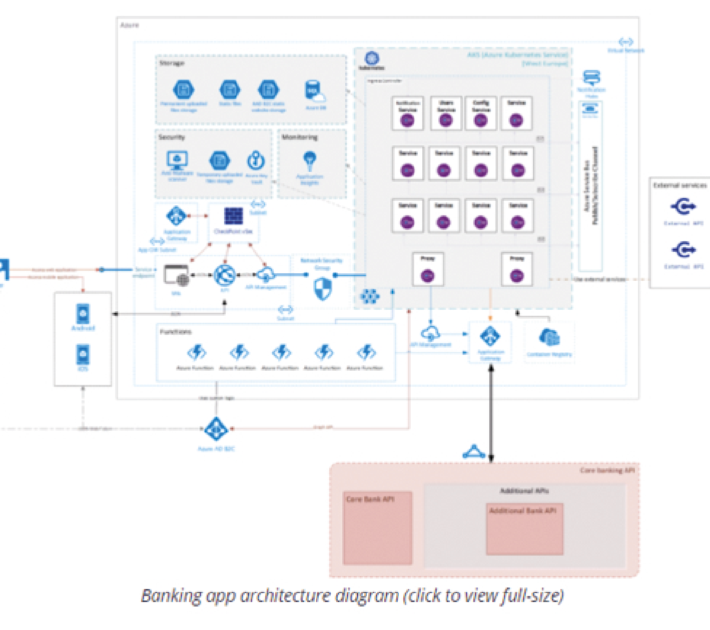
We used the Azure Kubernetes Service together with Azure Container Registry **to manage and deploy microservices.** This also allowed us to build a resilient and highly available back end. Microservices are hidden behind Azure API Management service which is a central gateway to access back end APIs.

Access to the solution is secured by **Azure Active Directory B2C,** so only authenticated users can use the functionalities available in web and mobile applications.

We're developing the web app using Angular and hosting it in the **Azure Web App** service. Mobile applications, Android and iOS, are being developed natively using Java and Swift respectively.

Important information related to user data will be securely stored in the **Azure SQL database.** Finally, all files that a user uploads to the application, will be sent to **Azure Blob Storage.**

Below you'll find the architecture diagram. As you can see, the solution is quite extensive.



Banking app architecture diagram (click to view full-size)

A product like this takes some time to build, but once it is complete, it will provide users with many useful functionalities and will make it easier for them to **manage their funds.**

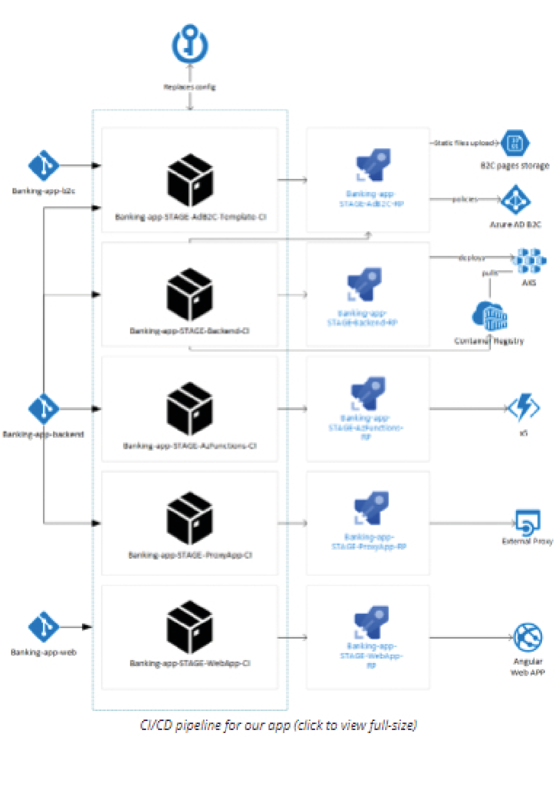
If you would like to find out more about the solutions we implemented or are implementing right now, then keep an eye on our blog. Soon, there will be a dedicated article where I will provide more details.

DevOps required!

As we saw above, such a multifaceted solution requires efficient management of adding new features and deployments. This is where DevOps practices come in. I will not discuss all of them here, but I encourage you to read my series about DevOps starting with **this article.**

What is important to mention here is the fact that having more than 10 microservices in a solution requires an efficient deployment process. For this reason, we use **Azure DevOps.** It is a tool that provides developers with services to support teams in planning work, collaborating on code development, and building and deploying applications.

Below you'll find the Continuous Integration and Continuous Delivery (CI/CD) diagram for the solution I described above.



CI/CD pipeline for our app (click to view full-size)

If you'd like to know more about the app or DevOps in general, **just get in touch** and I'll happily talk it over with you!

Key takeaways:

1. The PSD2 directive makes it possible to aggregate data from different bank accounts using APIs.
2. The Azure cloud provides services like Azure Kubernetes Service and Azure Container Registry for building microservice-based solutions.
3. The pillars of a secure modern banking application include authorization, data confidentiality, and encrypted connections and assets.

Daniel Krzyczkowski

Principal Software Engineer

My name is Daniel and I'm passionate about Microsoft technologies. I love learning new things about cloud and mobile development. On this blog, I will try to provide interesting articles about innovative technologies!