OutSystems architecture

The underlying OutSystems architecture is not apparent to the end-users of the apps, the developers creating them, or even the operations teams managing them. This article explains the architecture and how it maximizes the performance and security of applications, provides openness to continuous delivery processes, supports continuous innovation. It also demonstrates how a standard architecture is the only way to fully ensure no lock-in to a proprietary solution.

Platform Server

OutSystems Platform Server is the core of OutSystems. It is a server component that complements a standard web application server stack with a set of services.

OutSystems doesn’t use a proprietary runtime engine. All applications rely on standard architectures and frameworks.
OutSystems Platform Server is what makes this possible. It takes care of all the steps required to generate, optimize, compile, and deploy applications into a standard web application server, or package into a native mobile app, and has a set of specialized services under the hood:

- **Code generator**: This takes the application modeled in the visual editor and generates, compiles and assembles standard applications, secured and optimized for performance, running on top of standard technologies.

- **Deployment services**: These deploy the generated applications to a standard web application server, ensuring that your application is consistently installed on each front-end of your server farm / cloud infrastructure.

- **Application services**: These manage the execution of scheduled batch jobs, and provide asynchronous logging services to store events like errors, audits and performance metrics.

- **Mobile apps build service**: This packages the compiled mobile app into a native mobile app that is ready for distribution, either through the public or private stores or through Mobile Device Management (MDM) or Mobile Application Management (MAM) tools.

## Code generator

This compiler takes the application model that was developed in the visual editor and generates all the native application components, ready to be deployed to an application server.
When the code generator retrieves the application model, it:

- Runs a deep graph analysis on the application model. This graph is used to check for external dependencies and apply optimizations to the application.

- Checks for optimizations that can be done when generating the code. For example, if an application has a query to fetch data from the database, but the result of the query is never used in the application, the optimizer flags the query to be removed.

- Generates integration proxies to isolate an application from changes made to external services.

- Generates native code for all the layers in the application stack: from the SQL to interact with the database and the backend business logic services, to the JavaScript components in the front-end.

- Bundles session management, authentication, and configurations to the application.

- Adds logging and monitoring capabilities to the application, so that developers can check for errors, audits, and performance metrics at runtime.

- Generates the build scripts to upgrade the database model and to deploy the application.
Deployment services

The deployment services deploy the generated application components to an application server and ensure the application is consistently installed on each front-end server of an organization’s server farm.

There are two different deployment components. The deployment service deploys a .NET application on a specific front-end server. The deployment controller coordinates the deployment for all front-end servers as follows:

- The deployment controller uses the OutSystems code generator to compile the application models.
- Once the application is compiled, it sends the code to the deployment service on each front-end server. The deployment controller then monitors the deployment to ensure the application was successfully installed.
- If all deployments are successful, the deployment controller updates the runtime database to match the data models that were created. If not, the deployment is canceled, and the old application version continues running without changes.

Application services

OutSystems Platform Server also has services for managing the applications at runtime. There are two different services for this:
• **Scheduler service**: This service manages the execution of scheduled tasks.

• **Log service**: This service manages the error, audit and performance events of the applications.

## Scheduler service

OutSystems Platform Server supports the execution of scheduled batch jobs in applications. At runtime, the scheduling service retrieves the next pending job from the metadata database and executes it.

Batch tasks are designed for horizontal scalability so they are distributed across all the front-end servers.

## Log service

Applications are automatically instrumented to create error, audit, and performance logs. At runtime, these metrics are captured asynchronously, so that the performance of applications is not compromised while they create a constant stream of metrics.
The log service runs at a lower priority so that it doesn't compete with the normal processing of applications. At runtime, when an application generates an error, audit, or performance event:

- The event is written to a standard message queue on the front-end server.
- The log service running on the front-end server asynchronously retrieves the events from the message queue and adds them to the logs database.
- The events can then be leveraged by the monitoring consoles to display errors, audits, or performance data. They can also be sent to external systems by using available APIs.

### Mobile apps build service

The Mobile Apps Build Service (MABS) is the platform service responsible for packaging a mobile app into a native app package ready to be installed on mobile devices. MABS is designed to be a highly scalable service that completely abstracts the complexity of compiling, packaging, and signing a native app for iOS, Android, or both, so that development teams get a streamlined experience of building a mobile app with 1-click from the back-end to native front-end.
Each request to MABS is queued and processed in parallel in an isolated sandbox to ensure that information is always secure. All communication with MABS is done via the Platform Server and through HTTPS, and only the necessary information to execute the build operation is sent to MABS. This means that no backend logic information is sent. MABS stores all information with encryption and only for the minimum necessary time to complete the process.

The build process that MABS executes is:

1. Create a new project in an OutSystems template that provides the base native app structure and default native capabilities.
2. Process app resources (icons, splash screens, configurations).
3. Set the mobile platform: iOS or Android.
4. Add native plugins.
5. Process keys to sign the app.

When the mobile package is created, we delete all your information, collect the application package, and return it to you.