# Overview

This document will give an overview of the logical and physical architecture of the EOX Vantage Business Operating System Platform. The Platform is made up of server and client; with client being either web or mobile.

## Requirements

Offer a platform that offers various key services to build business applications.

Ability to deploy and manage applications that have varied functionality and purposes.

Support for multi-tenanting and enterprise installation

Ability to integrate with 3rd party applications

## Approach

Considering the key requirements it is proposed to use a microservices based architecture with defined APIs (REST, components and integration end points) exposed for use by client and application(s). Some of the key benefits in this approach are -

**Loose coupling** between different parts of the system offering high extensibility and maintainability

Each Application can be **developed and managed independently** without affecting other apps and functionality in the system

**Clear separation of server and client** gives the ability to support multiple client views spanning over different devices to even having customized UI.

Ability for **3rd Party developers** to develop applications for the EOX Vantage platform. Leading to a marketplace of rich variety of applications solving various business needs.

**An eventually consistent model** for data storage will be used for data storage. This approach will give us large scale scalability.

The micro-services based approach will give us flexibility to have **separate teams for each application**. Isolating each other and also the platform from the rest.

We can use **best of breed technology** across the landscape without limiting ourselves to one stack.

This approach will have the following drawbacks -

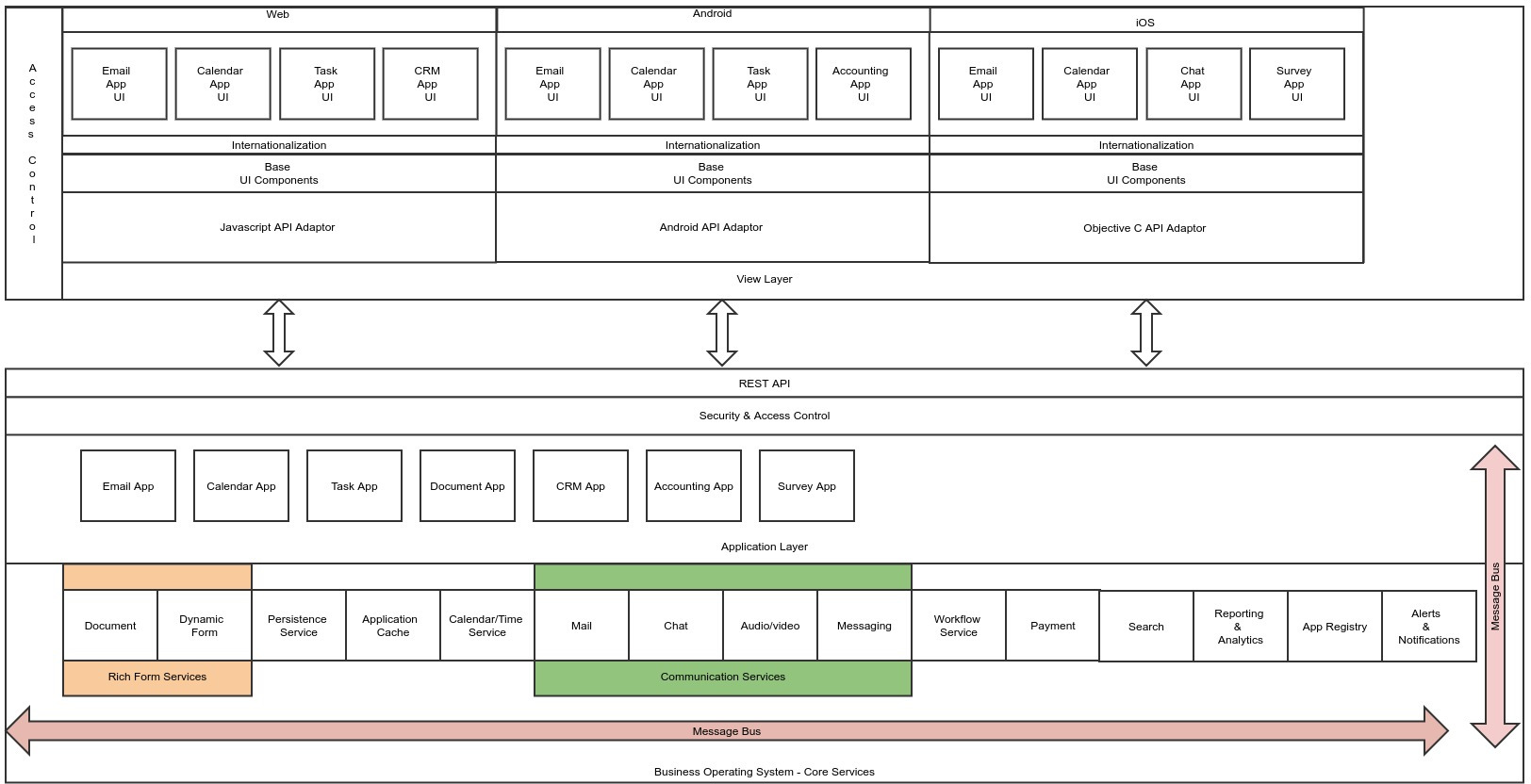
With micro-services approach would mean the **application deployment management will be more complex** and this can be overcome using automation and usage of proper tooling.

With the usage of eventually consistent model there is **no ACID consistency** across the data model. This will require proper asynchronous callbacks and handshakes to ensure corrections to enable eventual consistency.

We would **need developers across multiple stacks** to handle different technologies that are chosen. This problem can be managed by selecting those technologies for which we can easily find skilled people..

# Logical Architecture

The diagram below gives a logical architecture of the Enterprise Operating System.



## Enterprise Operating System (EOS)

## The Enterprise Operating System consists of libraries and functionality that offer the core capabilities required for building Business Applications. The APIs for these functionalities are exposed as REST or in-memory APIs which can be used to build & deploy different applications.

The Enterprise Operating System has many services and is not limited to the ones listed in the diagram and described below, this is only a representative list and can include others too.

### Rich Form Services

The Rich form service exposes the capability to represent complex dynamic content as forms or documents that are structured or unstructured.

### Persistence Services

The Persistence Service provides the capability for storing data on RDBMS, NoSql, File Systems or other kinds of data storage.

#### RDBMS Persistence

#### Each App can define its own set of tables, to avoid table name conflicts and also to ensure that access to data is restricted to the applications data following approaches can be taken.

##### Application Specific Database

#### Using this approach every application is given its own database and the application has to ensure that the access control is managed within the application. This would mean that the applications have to be certified for implementing proper access control before it is made available for use. A database connection specific to the application database can be obtained from the persistence service API before performing any database queries.

#### This method keeps the persistence service simple, but requires thorough verification to avoid improper access control implementation within the application.

##### Account/Application Specific Database

#### In this approach a database specific to an account and application is created uniquely so that account specific data for any given application is segregated and the application does not have to deal with any account specific data access security restrictions. Though, it has to implement role based access restrictions as required. The Persistence API  should ensure the connection provided is specific to the account and application.

This method keeps the persistence service simple and also assures the account specific access control. Only drawback is, maintenance can be cumbersome as each individual database has to be explicitly handled in backups/restorations.

##### Complete Abstraction using Views

In this approach the persistence API abstracts the tables by providing Views that are filtered by account using the account context from the user request. All inserts are done through the persistence API and subsequent, updates and queries can be executed directly on the view. The persistence service provides account specific connections that will give access to the views filtering the data for that account.

This method the persistence service handles the creation of database tables for the application with account identifier and generate a view that is account specific so that the application can use the view to perform all write and read operations through the view.

This is the recommended approach.

#### File Persistence

Every application will be given a folder specific to an account so that all data pertaining to an account will be clearly separated in its own folder. The File Persistence API will have APIs that will ensure the account specific access is factored in so the application need not add any further checks.

#### NoSql Persistence

NoSql Persistence can be used by applications where required. Every application will be given an account specific database so that the access control is implicitly handled. The application when fetching the connection will be given a connection that works with the account’s database only.

### Application Cache

The Application Cache is used for storing frequently used objects to improve overall efficiency. When objects are put in cache they can have a TTL (Time To Live) setting so it is available from cache when accessed while it is available. When trying to fetch the object after its TTL period the application should load the object from the persistence store and cache it again. When the object goes stale after it has been updated then a message has to be sent to mark all the objects as stale so it is no more available in the cache.

### Calendar/Time Service

This service offers ability to schedule and track one-time or recurring events.

### Communication Services

The services offered under the communication head can be related to different means of communication channels like email, chat, video/audio conferencing, messaging etc.

### Workflow Service

This service exposes a very important aspect of businesses; the processes involved in running the business. The process automation is an important aspect of the Enterprise Operating System with ability to design custom processes with specific activities and rules determining the flow of the process and who can perform a certain activity. Activity specific interfaces enable the assigned user to perform the activity appropriately.

### Reporting and Analytic Services

This service exposes APIs for building reports and drill down analytics on data belonging to various applications to provide insights in the various aspect of business.

### App Registry

The App registry is used to register new applications for different accounts. When an application is installed/uninstalled in an account the life-cycle events are triggered which can be used by the application to initialize and cleanup respectively.

#### Application Life Cycle Services

Every Application can implement the initialize and destroy life cycle callback operations. These callbacks are invoked in the context of an account.

The initialization operation gives the application opportunity to perform account specific initializations like setting up database tables, adding application specific privileges and subscribing for messages/events that it would like to listen for.

The destroy operation can handle any cleanup of data and unsubscribing all the subscribed messages/events.

### Payment Services

This service exposes APIs that can be used for performing various online payment transactions that would be required for various business processes.

### Search

The Search service offers a means to index and query different kinds of data belonging to various applications.

### Alerts & Notifications

This offers APIs to register for specific alert and for broadcasting notifications. The alerts and notifications can be sent/displayed on screen or using the appropriate communication channel.

## Application Layer

This represents the layer of applications that expose specific functionality offering specific business value. Some applications like Email App, Calendar App, Chat etc. are part of the core EOX Vantage platform. Newer apps can be developed by 3rd parties as well using the rich set of APIs and capabilities offered by the EOX Vantage platform.

## Security & Access Control

The services of this layer ensure only authenticated and authorized users are given access to the features and data that they are allowed to access.

### Authentication

Support for multiple authentication models like JWT, OAuth should be supported. This should be configured at the account level.

### Feature Plans

Every account will have specific features enabled based on the plans the organization has subscribed for. This gives ability to switch on/off features at the account level.

### Roles and Privileges

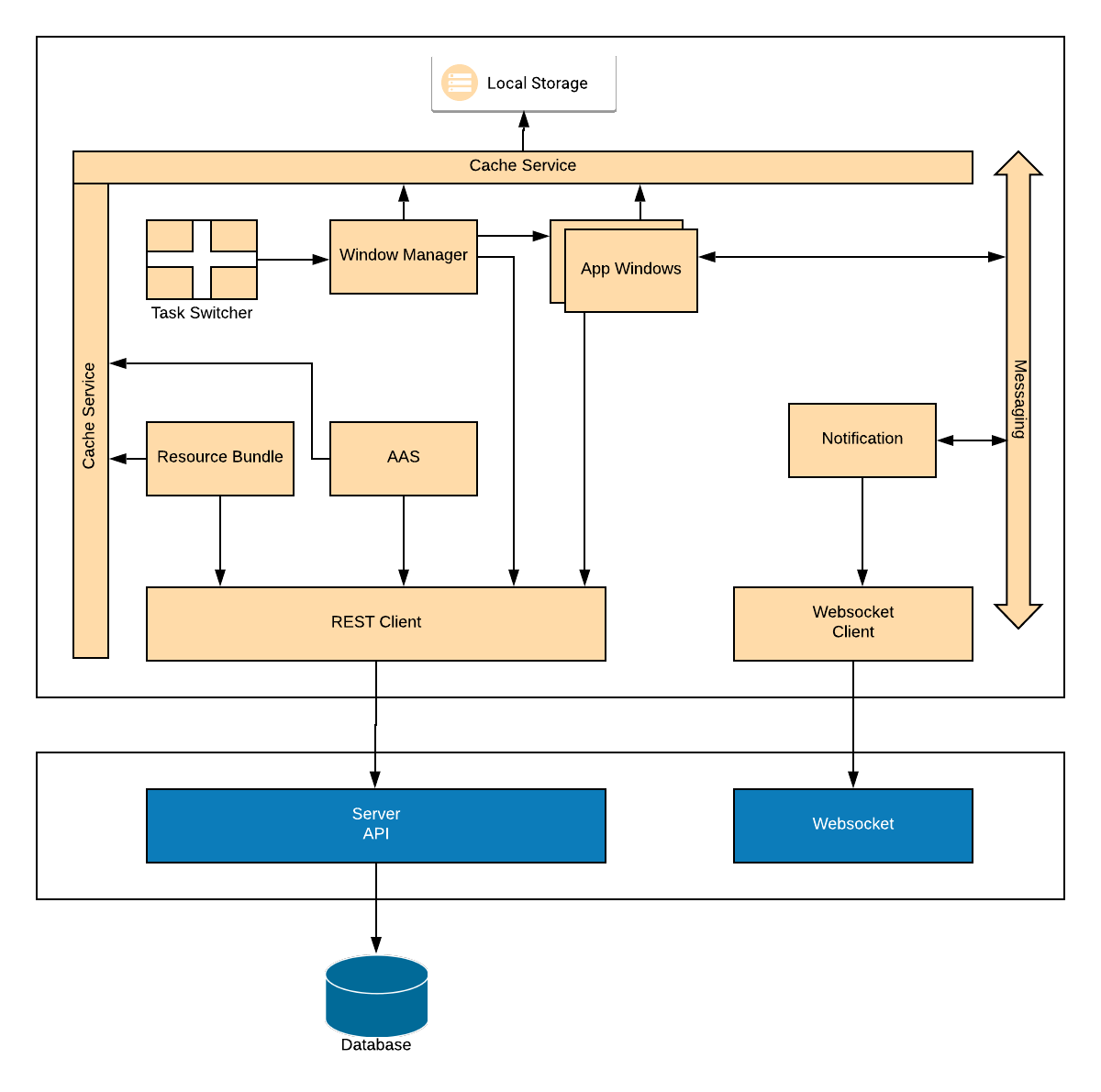
The Users in the system can be assigned to specific roles with each role having set of privileges corresponding to the various features and permissions assigned against each privilege like Read/Write/Create/Delete etc. Every app can add its own set of privileges which can be assigned to the various roles in the organization. This way the infrastructure is extensible for the newer apps that are added for an account.

## UI Layer

The UI layer offers a set of basic services for app registration, events & notifications. The applications are provided with a runtime environment that includes some important services like internationalization/localization, abstraction to the server side Enterprise Operating System APIs, a UI shell within which the applications can run.

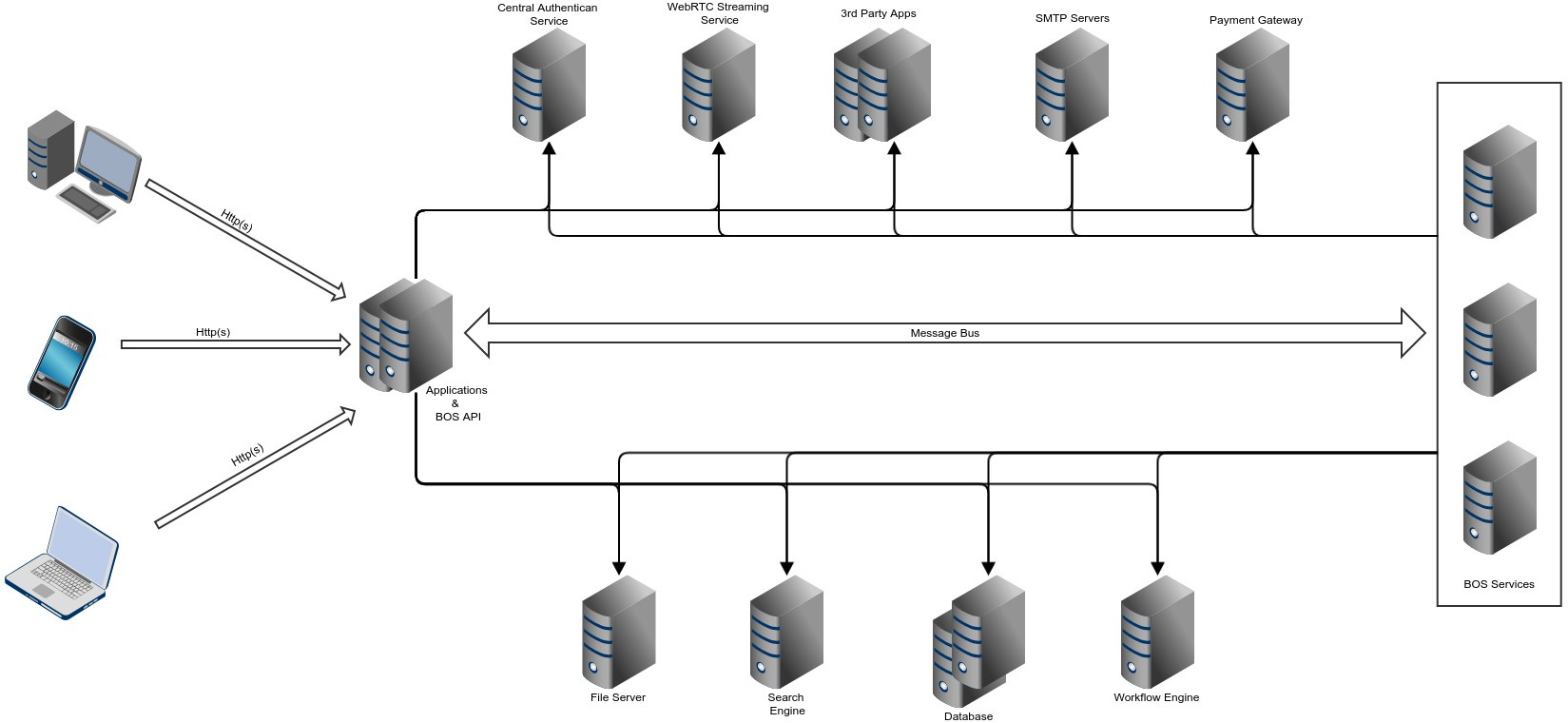
Client side libraries in various programming languages to suit the different client platforms like web and mobile.

In addition a rich set of UI components will be available to build applications.



# Physical Architecture

The diagram below shows the Physical architecture of the EOX Vantage platform. The various integrations presented in here are only representative and is not limited to only these.



## Applications and Enterprise Operating System API

The Applications and the Enterprise Operating System REST APIs are deployed on the web servers which is exposed to the internet for access to the client applications. These could be deployed on a cluster of servers.

## Enterprise Operating System Services

The various Enterprise Operating System services can be deployed on independent machines connected via a message bus. The Enterprise Operating System API will interact with the services by broadcasting messages through the message bus. This also gives ability to run certain processes asynchronously like search indexing, sending emails etc. This approach allows scalability of  individual services.

## Persistence Services

### Database

The Database servers can run one or more RDBMS or NoSql databases for different requirements. The database servers can be independently clustered for failover & redundancy.

### File Server

This is another form of storage where the requirement is for a file system storage. The file storage can be implemented with highly scalable file systems with high reliability and failover.

## Search Engine

This is a 3rd party search engine like Solr or Elastic-Search that can be used for indexing and searching various kinds of documents/content.

## Workflow Engine

The workflow engine is a 3rd party standards compliant workflow engine that could be used to implement the various business processes.

## Payment Gateway

There could be potentially integrations to one or more payment gateways for processing online payments.

## SMTP Servers

These could be SMTP servers belonging to various providers. Enterprise Operating System Email service will provide seamless integration with any standard compliance SMTP server for email communication.

## WebRTC Streaming

For supporting Audio/Video streaming services a standard compliance WebRTC service is required. The Enterprise Operating System service will abstract out the specifics for integrating with any available WebRTC service.

## Central Authentication Service

This provides a centralized authentication service for all the applications that run within the EOX Vantage platform or when integrations are provided for 3rd party applications with single sign-on capabilities.

## Message Bus

The message bus plays a very important role of connecting the various services in an asynchronous manner. It also makes the system loosely coupled and gives the flexibility to scale horizontally. The message bus will use a publish-subscribe model where services can subscribe for specific messages. All messages will be contextualized for an account, as a result all messages will always include account identifiers in its payload. When applications subscribe for a message they subscribe in the context of the account.

# Non-Functional Requirements

## Offline Access

Certain functionalities like email, documents, etc. should be made available in offline mode so the user experience is not hurt when there is a loss in connectivity. The client has to have the ability to store the changes on the client side and sync as and when the connectivity is restored in a seamless fashion..

## Security

All authentication will be based on username and password and usage of JWT tokens to identify the user context. All passwords will be hashed and saved.

All access over the network will use secured transport like HTTPS/SSH/SFTP etc.

The platform will impose strict restrictions to ensure tenant data isolation and prevent users from accessing data across the tenant boundary..skilled people

# Application Development

## Standalone Application

Any Standalone Applications can be integrated using a wrapper EOS application

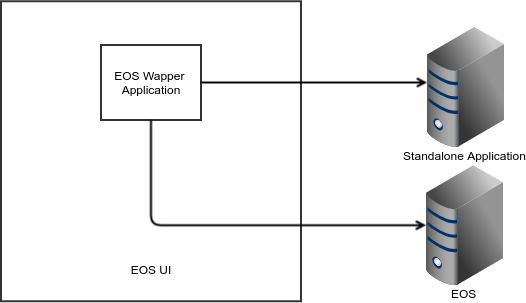
The Application can be implemented using any technology stack and can be deployed anywhere accessible over the internet

To integrate the standalone app with EOS the following needs to be taken care of

Single signon with EOS using the JWT token passed from the wrapper UI

APIs in the App to support data integration from EOS

Publish any data changes in the Application to be synced with EOS using the messaging channel.



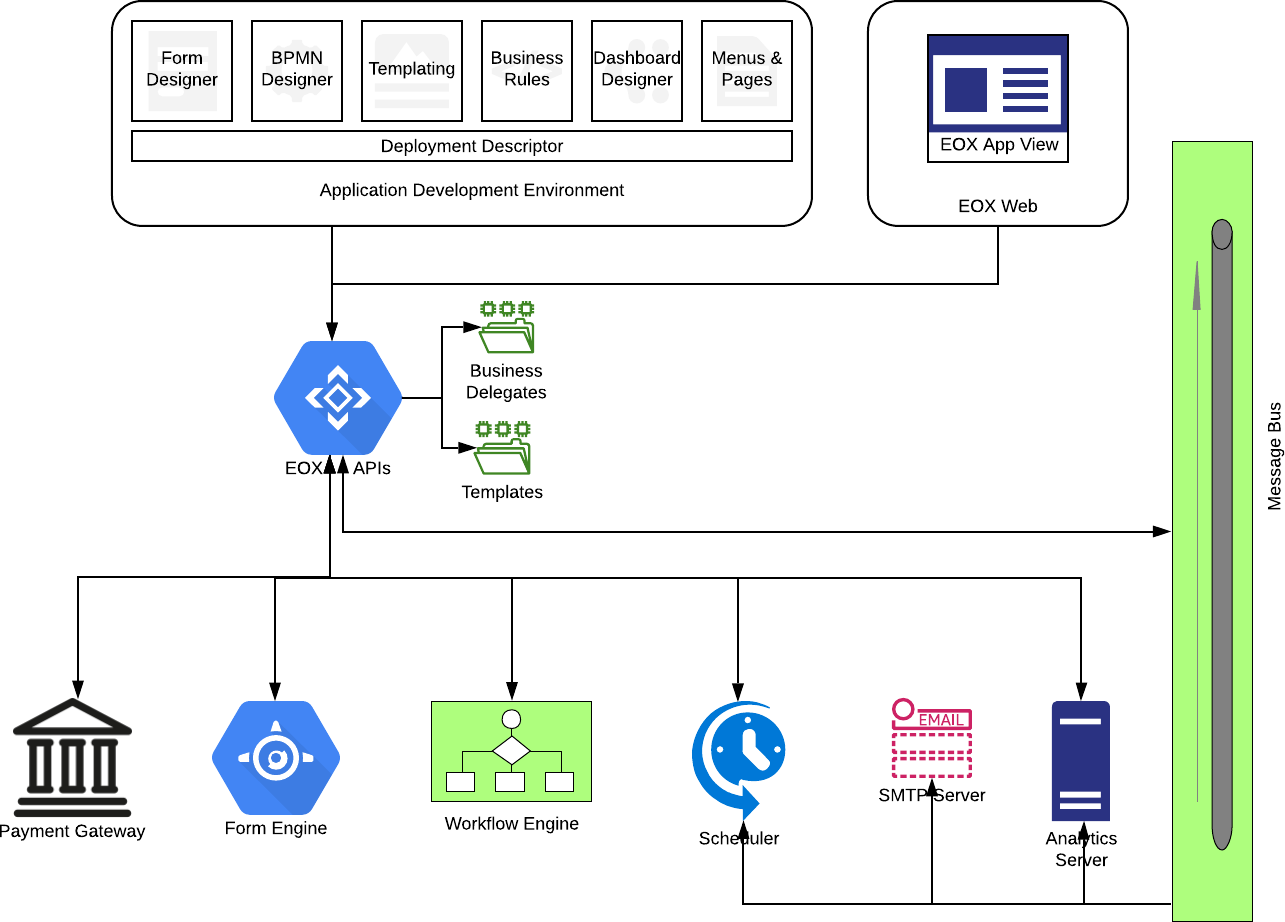
## EOX Application

EOX Application has to support the following capabilities

* Multi Tenancy
* Custom workflows
* Custom forms for workflows and data entry
* Generate documents
* Email notifications with custom templates
* Custom business logic
* Support definition of menus and pages
* Dashboards with support for Aggregates, charts and tables

### Overview

The diagram below illustrates the EOX application design

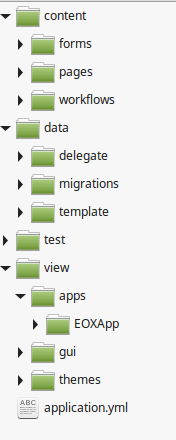


#### Application Development

The Application development has the following features

* Every application codebase can be managed independently
* Application is packaged in a specific structure with a descriptor defining the application
* Automated application deployment
* Definition of menus, pages and its content
* Workflow builder that uses BPMN 2.x
* Form designer to design custom forms and associate with the workflow activities
* Design of templates for email and pdf documents
* Develop and Test custom business rules with support for database access
* Dashboard designer to define custom dashboards
* Define UI theme for the Application and also develop custom UI components that can be embedded in the page
* Define privileges and access control for menus/pages

The folder structure for the application is as follows



The various aspects of application development are as follows

##### Form Designer

The Form designer provides the following capabilities

* Support rich set of components with client side validations
* Support for hooks to invoke server APIs

##### Workflow Designer

The workflow designer provides the following capabilities

* Standards based BPMN workflow definition

##### Templating

A Templating mechanism is required to be able to define templates for documents like invoices, policies that needs to be generated and issued as part of the business process

##### Business Rules

Any business application must have ability to implement custom business rules. This can be done using any programming language or through rule engines.

##### Dashboard Designer

The Dashboard designer provides the following capabilities

* HTML based editor with ability to embed analytics in a rich document format
* Support for Aggregates, formulas, charts and tables

##### Menus & Pages

The Application view can be configured with custom menus and page designs with support for access control.

##### Deployment Descriptor

The deployment descriptor contains the details about the application that is required to deploy the application in the EOS server. It defines the following

* Application name, identifier
* Roles and privileges
* Group
* Menus and pages linked to them
* Entities, forms and related workflows
* Scheduled Jobs to be setup

#### EOX App View

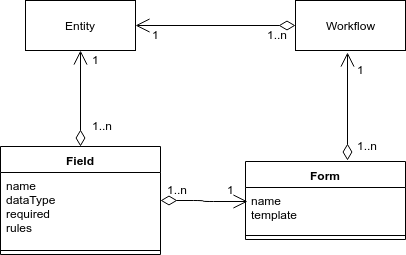
The EOX Application View is data driven with ability to render the application menu and pages as defined in the metadata. The following features are supported

* Multi-level menus
* Access Control
* Pages that can render the following elements
  + Forms
  + Tables with ability to
    - load data from APIs
    - support row level actions
    - support actions
  + Static HTML content
  + Dashboards
  + Document Viewer
  + Custom Components

### Design

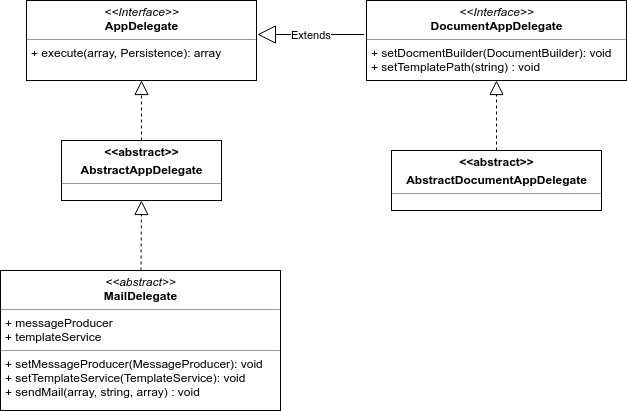
#### The Entity Model

The Entity model defines the relationships between the key elements like workflow, form and fields. All of them belong to an Entity. The diagram below depicts the relationship between them.



#### Business Delegate Model

* The Business delegates are custom rules implemented using PHP.
* The delegates have access to an application specific SQL database.
* The database supports multi tenancy and automatic version migration.



#### EOX APIs

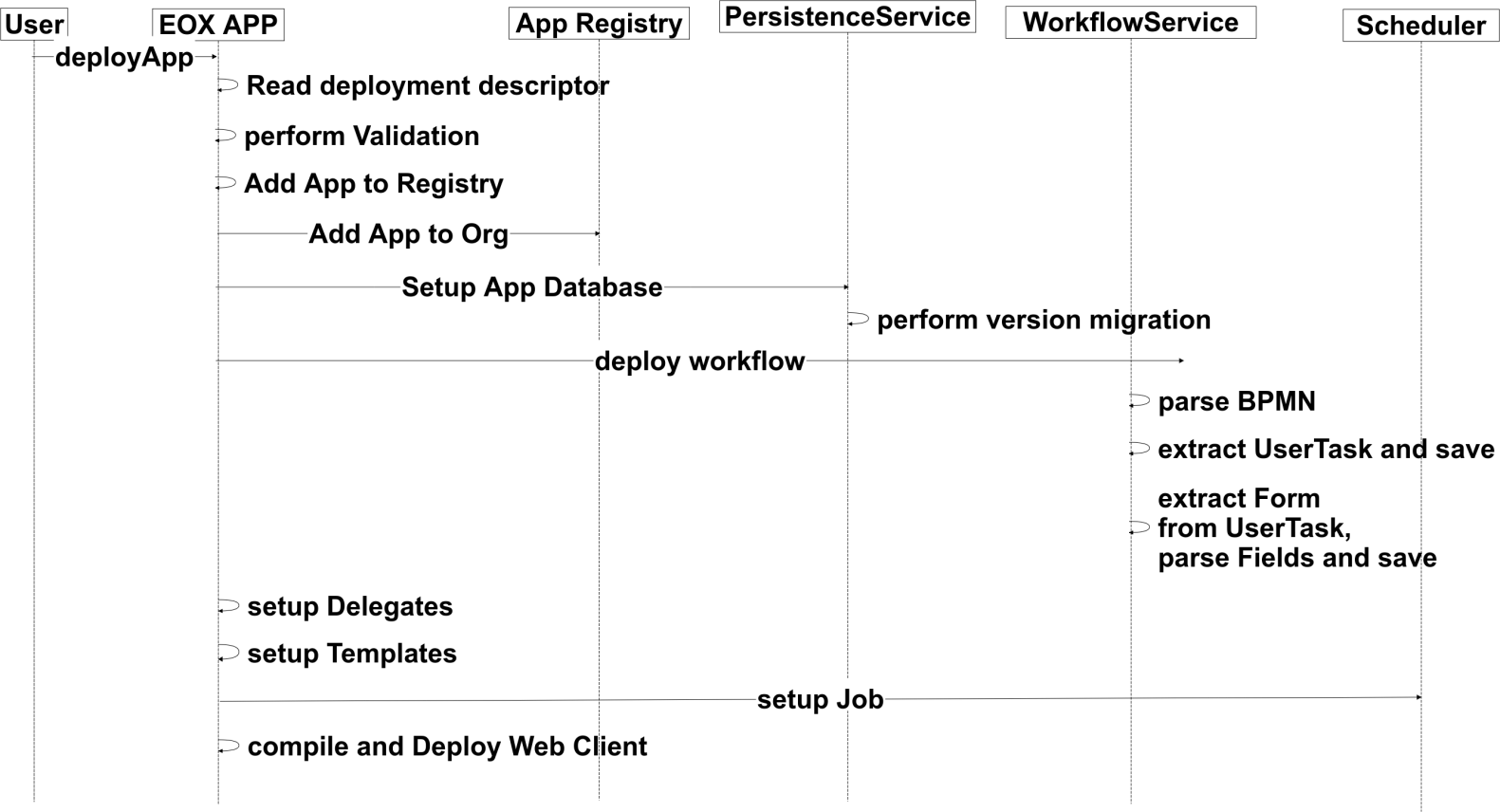
The EOX Apis offers a seamless way to interact with

* App – deploy, getMenu, getPage, getForm
* User – signup, signin
* Payment – initiate, process, updateStatus
* Workflow – deploy, submit, taskAssignments, claimTasks
* Email - send
* Analytics – executeQuery, createWidget, updateWidget, listWidget getWidget, saveQuery, getQuery, listQuery
* Scheduler – schedule, cancel
* Business delegate - execute
* Document generation – generate, get

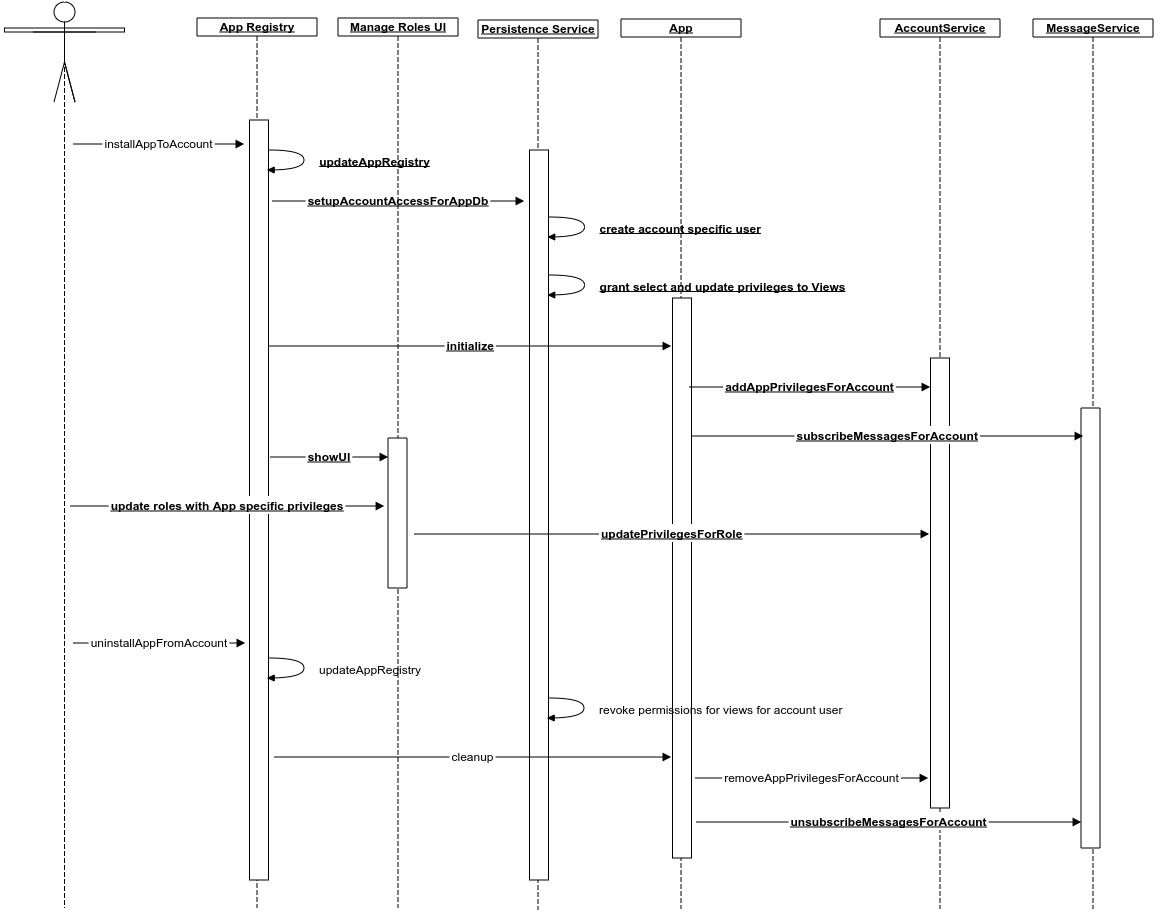
# Sample Flows

Some representational sample flows are given below to give a better understanding of some of the basic operations.

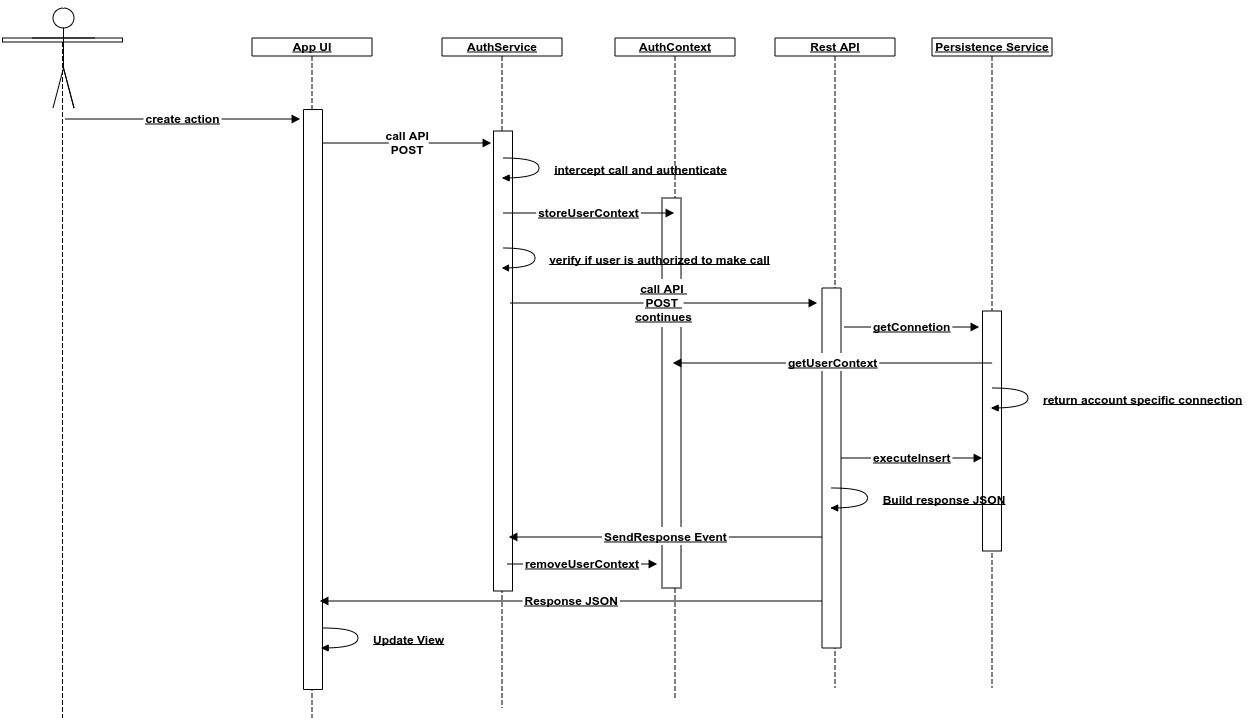
## App Deployment



## App Installation/Uninstallation for Account



**Application REST API Call (POST)**



# Technology Stack

|  |  |
| --- | --- |
| Operating System | Linux |
| Programming Languages | PHP 7.x, Java, Groovy, Javascript, ES6 |
| Frameworks | Zend 3.x, Quartz, Spring, ReactJS, OSJS, FormIO, AMCharts |
| Database | MySQL 5.x / PostgreSQL |
| Development Tools | Sublime Text, Visual Studio Code,Eclipse, Docker, Git, Jenkins, Gradle, BPMN modeler |
| Workflow Engine | Camunda 7.x |
| Messaging | Active MQ |
| Integration Patterns | Apache Camel |
| SMTP | Postfix |
| Analytics | Elastic 7.x |
| Form Engine | FormIO |
| Templating | Smarty |
| PDF Document Generation | Snappy, WkHtmlToPdf |
| Scheduler | Quartz |
| Charting | AMCharts |