CLOUDSAT ERP

Solution Design Document

Architecture of CloudSatERP

Introduction

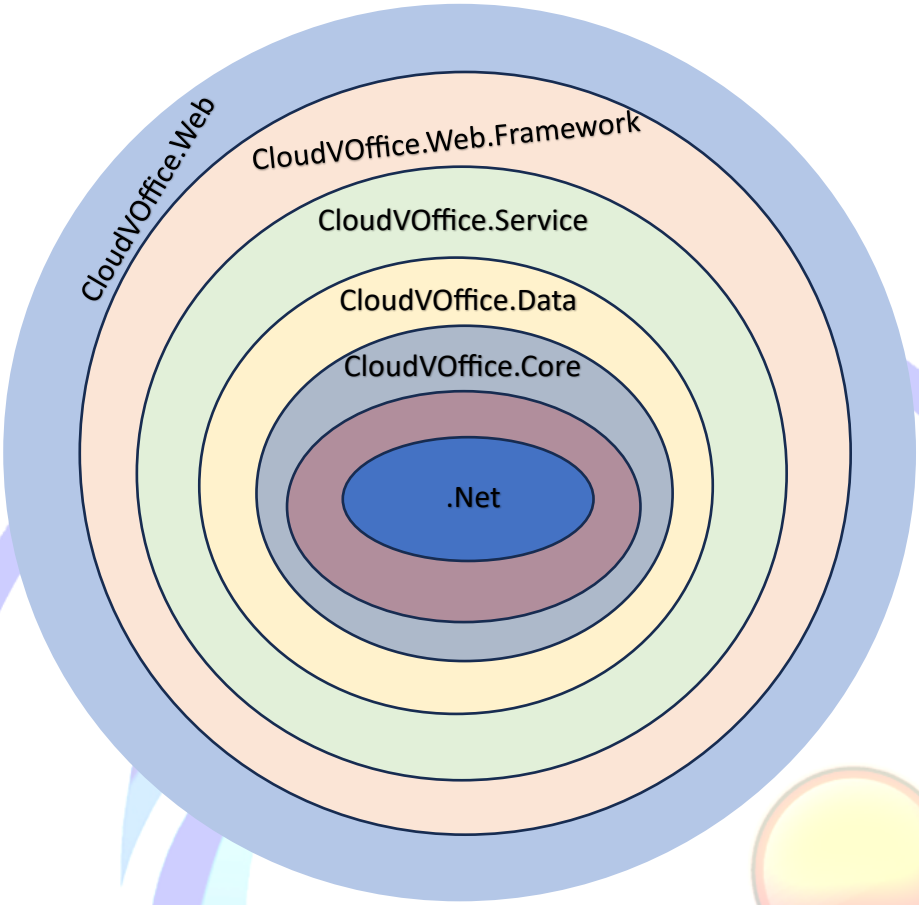
CloudSat is a highly customizable and flexible, full-featured ERP solution. Which is built on top of Microsoft `ASP.NET Core` framework. CloudSat ERP is always up to date with the latest technology and follows best practices.

Overview

This document provides a comprehensive architectural overview of the CLOUDSAT ERP system. This documentation is targeted at new Cloudsat erp developers. In this documentation, we will be exploring each project in Cloudsat erp solution, dependencies between those projects, etc.

Overview of CloudSat ERP Architecture

CloudSat Erp is one of the most popular and successful Dot NET based ERP solutions. The success of CLOUDSAT ERP is not only because it contains most of the features required by a modern ERP solution, out of the box and its UI is highly customizable and User friendly, but also because the CLOUDSAT ERP solution is equally organized and developer-friendly. The main strength of CLOUDSAT ERP is its flexible, extendable architecture and well-organized source code. The erp architecture is very close to the onion architecture. Which is mainly focused on controlling the code coupling. According to this architecture, all code can depend on layers more central, but code cannot depend on layers further out from the core. In other words, all coupling is toward the center.



This means projects can only have a dependency on the other projects that reside inward from the current project. For example, if you see the above diagram CloudVOffice.Data project can depend on the CloudVOffice.Core project and can have CloudVOffice.Core as a dependency, but CloudVOffice.Core cannot depend on CloudVOffice.Data similarly CloudVOffice.Services can have CloudVOffice.Data and CloudVOffice.Core as it's dependencies but neither CloudVOffice.Core nor CloudVOffice.Data can have CloudVOffice.Service as their dependency. This means projects can only have another project as a dependency only if it resides inward or more center from the layer the current project resides. Which is the key to code decoupling. The main advantage of this approach/architecture is that, now if we want to test the Application Core then we can do so even if we do not have any UI for our application. Because the Application core does not depend on UI layer. Or we can change our UI framework from Razor view engine and JQuery to Angular or React or Vue without affecting our Application Core and can use the same core to build Mobile Applications or Desktop Applications. All of this without changing a bit of code in our Application Core.

Projects in CloudSatERP Solution

Application Core

This is the innermost layer in the CloudVOffice architecture. It is the core of the application. All the data access logic and business logic classes reside inside this layer. In the CloudVOffice solution, we can find all of the projects for this layer inside the "Libraries" directory. This layer contains three projects.

CloudVOffice.Core

This project contains a set of core classes for CloudSaterp. This project is at the center of the architecture and does not have any dependencies on other projects in the solution. This project contains core classes that are shared with the entire solution like Domain Entities, Caching, Events, and other Helper classes**.**

CloudVOffice.Data

The CloudVOffice.Data project depends on the CloudVOffice.Core project, which is in the middle of the architecture. It does not depend on any other project in the solution. The CloudVOffice.Data project has classes and functions for reading and writing to a database or other data store. It separates data-access logic from business objects.

CloudVOffice.Service

The CloudVOffice.Services project is the outermost layer of the Application Core in CloudSatERP architecture. It depends on the other two projects in the Application Core. It has core services, business logic, validations, and calculations for the data. Some people call it the Business Access Layer (BAL) or Business Logic Layer (BLL). It serves as the facade for all the other layers. It has service classes and uses a repository pattern to expose its features and functionalities. This approach decouples the core from the other layer outside the core. It also prevents or reduces code changes in the other layers if the logic for the Application Core changes. This approach is ideal for dependency injection.

UI Layer

This layer resides outside the "Application Core". In the CloudSatERP solution, we can find all of the projects for this layer inside the "Presentation" directory. All the presentation logic and UI resides in this layer. This is the layer where the UI which users can interact with, presents. In CloudVOffice, this layer breaks into two other layers.

CloudVOffice.Web.Framework

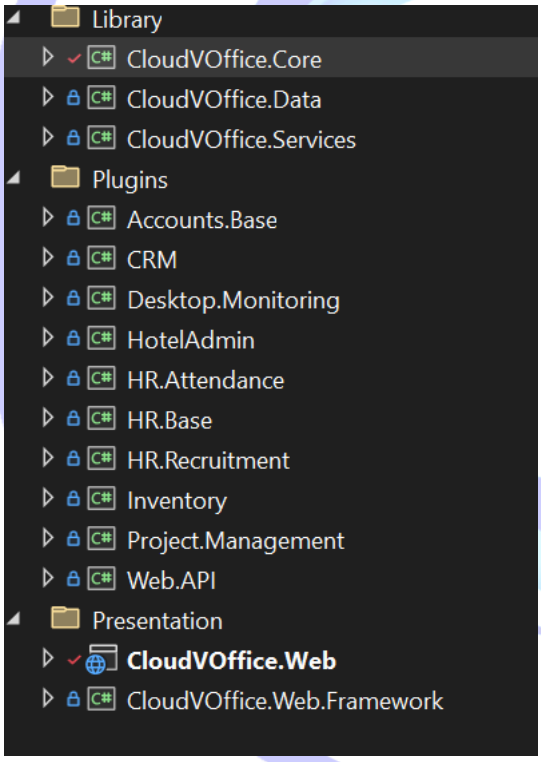
The CloudVOffice.Web.Framework project is the inner layer of the Presentation layer and depends on the Application Core layer. It is a class library project that serves as a framework for the Presentation layer. It has shared logic for both the CloudVOffice public website and the admin panel.

CloudVOffice.Web

The CloudVOffice.Web project is the outermost layer of the Presentation layer in CloudVOffice architecture. It has the front-end website user interface that users can interact with. It is an ASP.NET Core application that depends on CloudVOffice.Web.Framework and the Application Core. It uses CloudVOffice.Web.Framework for the shared logic between this and the Admin panel. It uses the Application Core for data access and manipulation.

Source code organization

This document is a developer's guide to the structure of the CloudVOffice solution and is intended to help new developers learn the source code. Projects and folders are listed in the order in which they are sorted in Visual Studio. We recommend that you open your CloudVOffice solution in Visual Studio and view projects and files as you read this document.



Most of the projects, directories, and files are named so that you can get a rough idea of their purpose. For example, you don't even have to look inside the project called Accounts.Base to guess what it does.

\Libraries\CloudVOffice.Core

The CloudVOffice.Core project contains a set of core classes for CloudVOffice, such as caching, events, helpers, and business objects (for example, Order and Customer entities).

\Libraries\CloudVOffice.Data

The CloudVOffice.Data project contains a set of classes and functions for reading from and writing to a database or other data store. The CloudVOffice.Data library helps separate data-access logic from your business objects. CloudVOffice uses the Linq2DB Code-First approach. Code-First allows a developer to define entities in the source code (all core entities are defined in the CloudVOffice.Core project), and then use Linq2DB and FluentMigrator to generate the database from the C# classes. That's why it's called Code-First. You can then query your objects using LINQ, which translates to SQL behind the scenes and is executed against the database. CloudVOffice uses Fluent API to fully customize the persistence mapping.

\Libraries\CloudVOffice.Services

This project contains a set of core services, business logic, validations, or calculations related to the data if needed. Some people call it Business Access Layer (BAL).

Projects in the \Plugins\ solution folder

Plugins is a Visual Studio solution folder that contains plugin projects. Physically it's located at the root of your solution. But plugins DLLs are automatically copied in the \Presentation\CloudVOffice.Web\Plugins directory which is used for already deployed plugins because the build output paths of all plugins are set to ..\..\Presentation\CloudVOffice.Web\Plugins\{Group}.{Name}. This allows plugins to contain some external files, such as static content (CSS or JS files) without having to copy files between projects to be able to run the project.

\Presentation\CloudVOffice.Web

`CloudVOffice.Web` is an MVC web application project, a presentation layer for the public store which also contains an administration panel included as an area. If you haven't used `ASP.NET` before, please find more info here. This is the application that you run. It is the startup project of the application.

`\Presentation\CloudVOffice.Web.Framework`

`CloudVOffice.Web.Framework` is a class library project containing some common presentation things for the `CloudVOffice.Web` project.