OutSystems における Architecture Canvas を用いたアーキテクチャ設計の考え方

tdi 情報技術開発株式会社

システム開発の生産性、保守性の向上やデジタルトランスフォーメーション推進の手段として注目されているローコード開発。中でも「OutSystems」は、アプリケーション開発の生産性だけではなく、開発基盤として、開発後の運用・保守における生産性の効率化にも貢献できる製品として注目されています。

当記事では、OutSystems のアーキテクチャ設計手法である「Architecture Canvas」についてご紹介いたします。

目次

1. Architecture Canvas とは

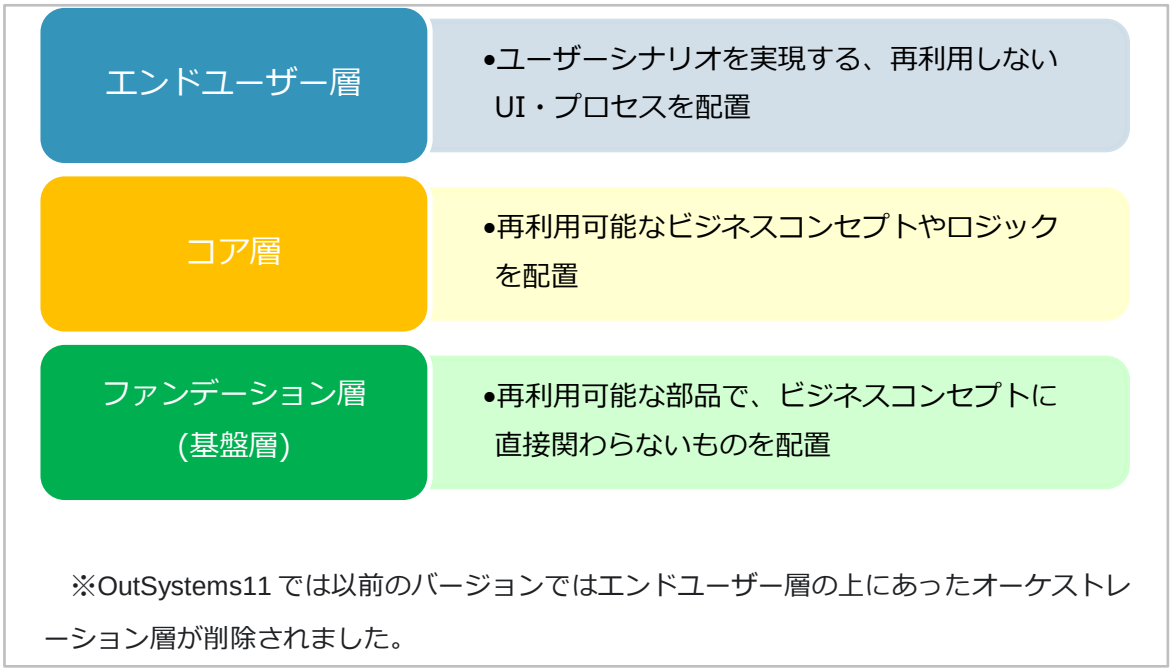
Architecture Canvas とは、OutSystems が推奨しているアーキテクチャ設計手法・図法のことです。OutSystems を利用してアプリケーションを作る際の指針の中で、Architecture Canvas はアーキテクチャ設計におけるベストプラクティスとして知られています。作成するモジュールの再利用化・保守性を向上させることを目的としたモジュール作成における方針をまとめたものとなっており、公式サイトでは以下のように説明されています。

"Architecture Canvas は、サービス指向アーキテクチャ（SOA）の設計を簡素化するための OutSystems のアーキテクチャツールです。これにより、共通モジュールを再利用する複数のアプリケーションを開発、保守している場合に、再利用可能な（マイクロ）サービスの適切な抽象化と個々の機能モジュールの適切な分離がしやすくなります。"

(※引用元：https://success.outsystems.com/ja-jp/Documentation/Best_Practices/Architecture/Designing_the_Architecture_of_Your_OutSystems_Applications/The_Architecture_Canvas)

2. Architecture Canvas の考え方

Architecture Canvas では、図 1 の 3 つのレイヤー(階層)の考え方に従って想定される機能を分割・分類することで、アプリケーションのアーキテクチャ設計を行います。



※OutSystems11 では以前のバージョンではエンドユーザー層の上にあったオーケストレーション層が削除されました。

図 1. Architecture Canvas のレイヤー

3. Architecture Canvas を用いたアーキテクチャ設計

ここでは、Architecture Canvas を用いたアーキテクチャ設計のプロセスについて説明します。

- 新規アプリケーションに Architecture Canvas を適用する

新規アプリケーションに Architecture Canvas を適用する方法について説明します。Architecture Canvas を用いた新規アプリケーションを開発する際は、図 2 の通り、各モジュールがアーキテクチャ構成のルールに違反しないようにモジュールを分割して設計を行う必要があります。

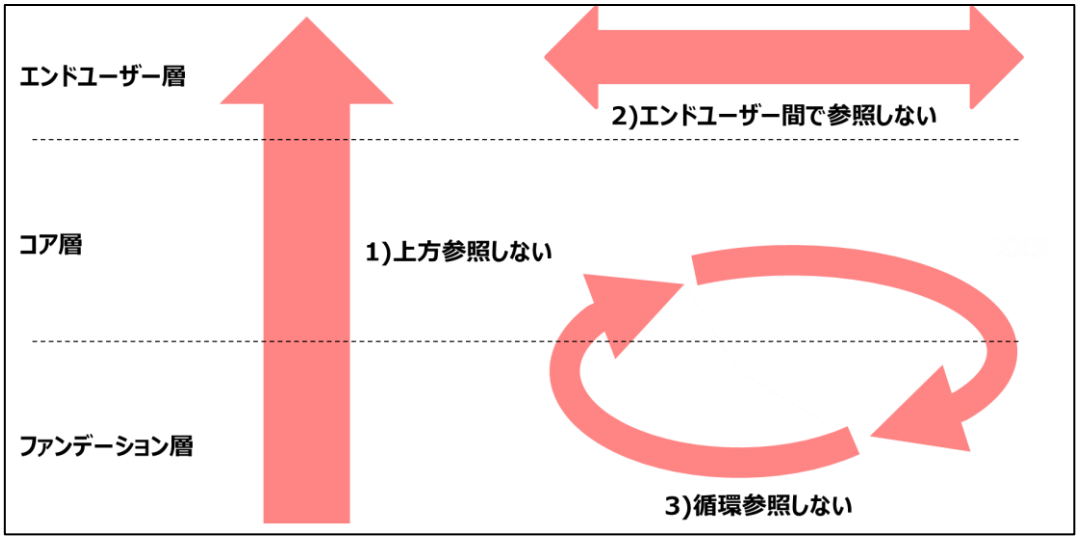


図 2.アーキテクチャ構成のルール

1) 上方参照をしない(例：コア層からエンドユーザー層への参照)
2) エンドユーザー層間でのサイドリファレンスをしない
3) コア層またはファンデーション層間での循環参照をしない

上記のルールを守りながらモジュール分割が完了したら、アーキテクチャ構成のルールに従って、デプロイする単位でモジュールをまとめてアプリケーション化していきます。アプリケーションは開発チーム別・顧客別・変更頻度別など、デプロイ運用がしやすい単位に分けることを意識するようにすると良いです。

- 既存アプリケーションに対して Architecture Canvas を適用する

OutSystems で構築されたアプリケーションに Architecture Canvas の考え方を適用する方法について説明します。OutSystems では、Forge で提供されている「Discovery ツール」を用いて、ルールに違反しているモジュールを検出することができます。Discovery ツールによって検出されたモジュール間の違反を、Architecture Canvas の設計ルールに従って解消することで、既存アプリケーションのアーキテクチャを Architecture Canvas の概念に沿った形にリファクタリングすることができます。

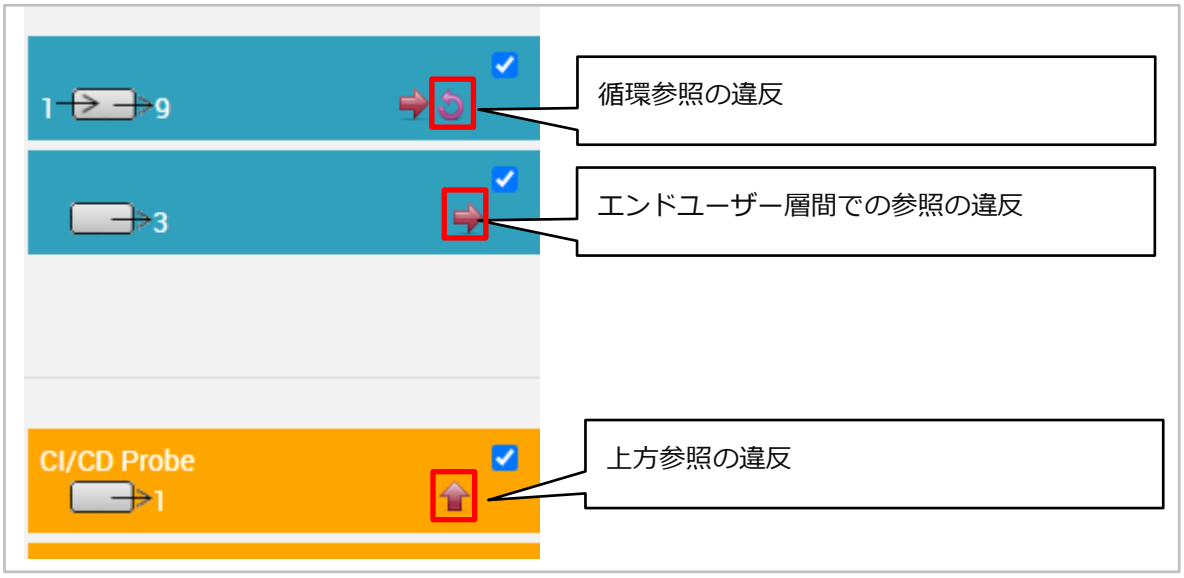


図 3. Discovery ツールによって検出されたアプリケーション間の違反 image

図 3 のように違反が発生していた場合、その違反を解決する手順としては、以下のようになります。

1.違反の解決は、最上位レイヤーから順に実施する

2.直接的な循環参照の違反を解決した後に、上方参照の違反を解決する

3.アプリケーションから参照されている数が多いモジュールの違反解決を優先する

4. Architecture Canvas の採用事例

Architecture Canvas を用いたアーキテクチャ設計の手法について、実プロジェクトでの採用例をご紹介します。

今回ご紹介するアプリケーションは、倉庫の在庫管理アプリケーションです。作成するアプリケーションの要件は、下記の通りです。

・ タブレットで在庫情報を登録・更新する
・ 在庫情報の登録は、商品の QR コードをスキャンして商品情報を読み込む
・ 登録・更新した在庫情報をサーバーにアップロードする
・ アップロードされた在庫情報は、Web 画面で確認や承認を行う
・ マスタ情報等は既存の外部 DB から取得する
・ 在庫確認情報は OutSystems 内の内部 DB で管理する

上記要件を満たすアプリケーションのアーキテクチャの設計手順について、順を追って説明していきます。

● 手順 1.要件の洗い出し

まず、本アプリケーションのプロセスとしては、倉庫内の在庫情報の登録・更新・参照処理、登録情報の承認処理が挙げられます。

要件の把握においては、各プロセスについて、UI/UX・ビジネス要件・連携機能の観点から、ユーザー目線で実現したい機能に落とし込みます。

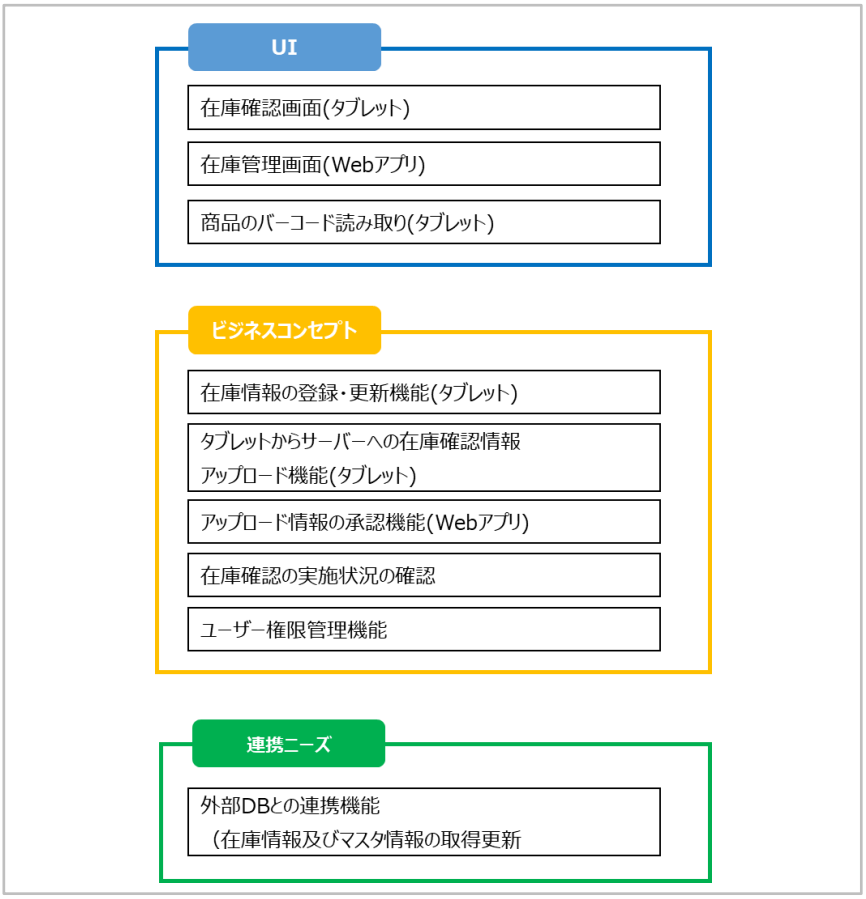


図 4.要件から洗い出された機能

● 手順 2. 洗い出された機能を Architecture Canvas で整理

ユーザー視点で洗い出した機能を、Architecture Canvas の各レイヤーに配置していきます。Architecture Canvas に配置する機能は、開発者視点で必要となる要件になります。

なお、複数のプロセスがある場合はそれらを統合する必要があります。

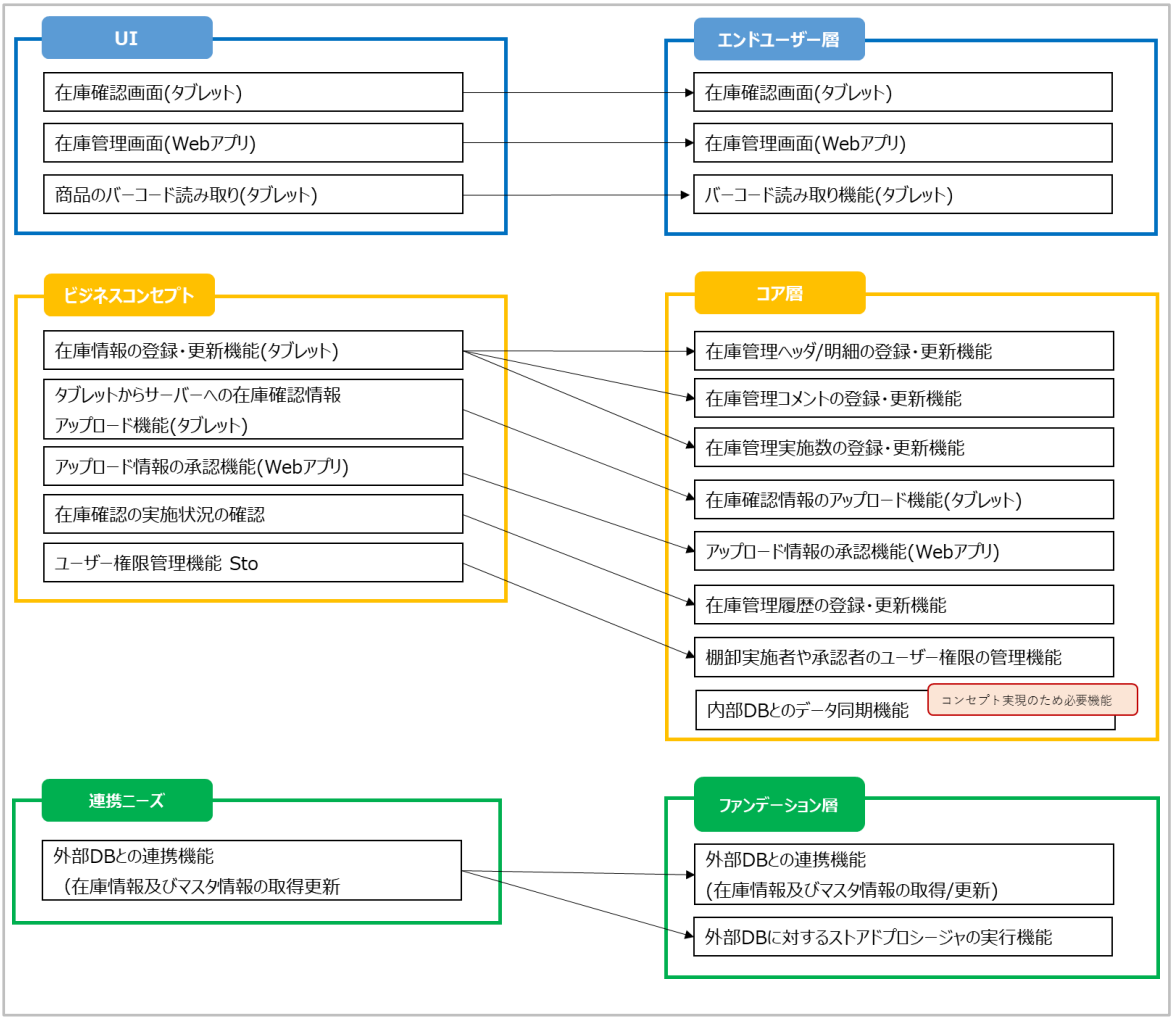


図 5.要件を Architecture Canvas に配置した際の対比図

● 手順 3.分割した機能を繋ぎ合わせモジュール化

整理した機能を繋ぎ合わせてモジュール化します。モジュール化する際には、以下の点に留意してモジュールを構成させます。

- 密接に関連している機能を結合する
- 複雑すぎる機能は結合しない
- ライフサイクルが異なる機能同士を結合しない

・ 再利用可能なロジックは、統合ロジックから分離する
・ 「3. Architecture Canvas を用いたアーキテクチャ設計」で記載した参照ルールに従う

今回ご紹介したアプリケーションをモジュール化した結果は、表 1 のようになります。

表 1.要件から分割した機能の構築モジュール、モジュール化観点の一覧

レイヤー	(図5で配置した)機能	構築するモジュール	モジュール化の観点
エンドユーザー層（再利用しないUI・プロセス）	在庫管理メニュー機能(タブレット)	StockTaking モジュール	開発方式(Mobile/Reactive Web)の違いによりモジュールを分割する
	バーコード読み込み機能(タブレット)		
	在庫管理メニュー機能(Webアプリ)	StockTakingWeb モジュール	
コア層（再利用可能なビジネスコンセプト・ロジック）	ユーザー権限管理機能	StockTaking_Users モジュール	棚卸実施者や承認者のユーザー権限の管理機能をモジュール分割する
	在庫管理ヘッダ登録・更新機能	StockTaking_CS モジュール	密接に関連しているコンセプト（在庫管理の情報を登録・更新する機能）を結合してモジュールとして構築する
	在庫管理履歴登録・更新機能		
	在庫管理コメント登録・更新機能		
	在庫管理実施数登録・更新機能		
	DBとのデータ同期機能	StockTaking_Sync モジュール	ベストプラクティス※に沿い、データ同期のモジュールを分割する
ファンデーション層（再利用可能な部品、等）	外部DBとの連携機能	Data_ISモジュール	外部DBとの連携を実現するための機能をモジュール化する
	外部DBに対するストアドプロシージャの実行機能	Data_SPモジュール	Data_ISモジュールとライフサイクルが異なるためモジュール分割
	バーコードライブラリ	Barcode_Lib モジュール	Forge機能としてモジュール化する
	インプットマスクプラグイン	InputMaskMobile モジュール	Forge機能としてモジュール化する
	タブレット用テーマ	StockTaking_Theme モジュール	ベストプラクティス※に沿い、タブレット/Webアプリのレイアウトに関する共通テーマとしてそれぞれ分割する
	Webアプリ用テーマ	StockTakingWeb_Theme モジュール	
	カレンダープラグイン	SimpleMonthCalendar モジュール	Forge機能としてモジュール化する

- 手順 4.モジュールをデプロイ単位で束ねる

モジュール作成と同様、「3.Architecture Canvas を用いたアーキテクチャ設計」で記載したモジュールの参照ルールに留意しながら、モジュールを束ねアプリケーション化します。OutSystems では、デプロイをアプリケーション単位で施します。そのため、デプロイ管理のしやすい単位、もしくは、アプリケーションを利用する単位で分割を行うことがベストプラクティスとなります。例えば、利用者が管理者と一般ユーザーでアプリケーションが分かれる場合は、管理者と一般ユーザーでアプリケーションを分割します。

今回取り上げている在庫管理アプリケーションにおけるアプリケーション化は、表 2 の観点で行います。

表 2.在庫管理アプリケーションにおけるアプリケーション化の手順と留意点、作業結果

No	モジュール名	アプリケーション名	観点・留意点
1	・StockTaking ・StockTaking_Theme ・InputMaskMobile ・Barcode_Lib	StockTaking	・タブレットアプリとWebアプリでは作り方が異なるため、 別アプリケーションとして管理する。 ・Themeモジュール、及び、そのアプリケーションにおいてのみ 利用されるモジュールをそれぞれのアプリケーション内に配置。
2	・StockTakingWeb ・StockTakingWeb_Theme ・SimpleMonthCalendar	StockTakingWeb	
3	・StockTaking_CS ・StockTaking_Users	StockTaking Core Service	StockTaking、StockTakingWebアプリケーションの双方から 利用されるビジネスロジックモジュール群をアプリケーションとして 分割し、再利用化を図る。
4	・StockTaking_Sync	StockTaking Data Sync CS	・コア層モジュールのデータ同期モジュール、ファンデーション層の 外部DBとの連携モジュールをさらに分割してアプリケーション化。 ※デプロイ単位の最適化と保守性の向上も図るため
5	・Data_IS ・Data_SP	StockTaking DB CS	

以上で、アーキテクチャ設計は完了となります。完成したアーキテクチャ図が図 6 になります。

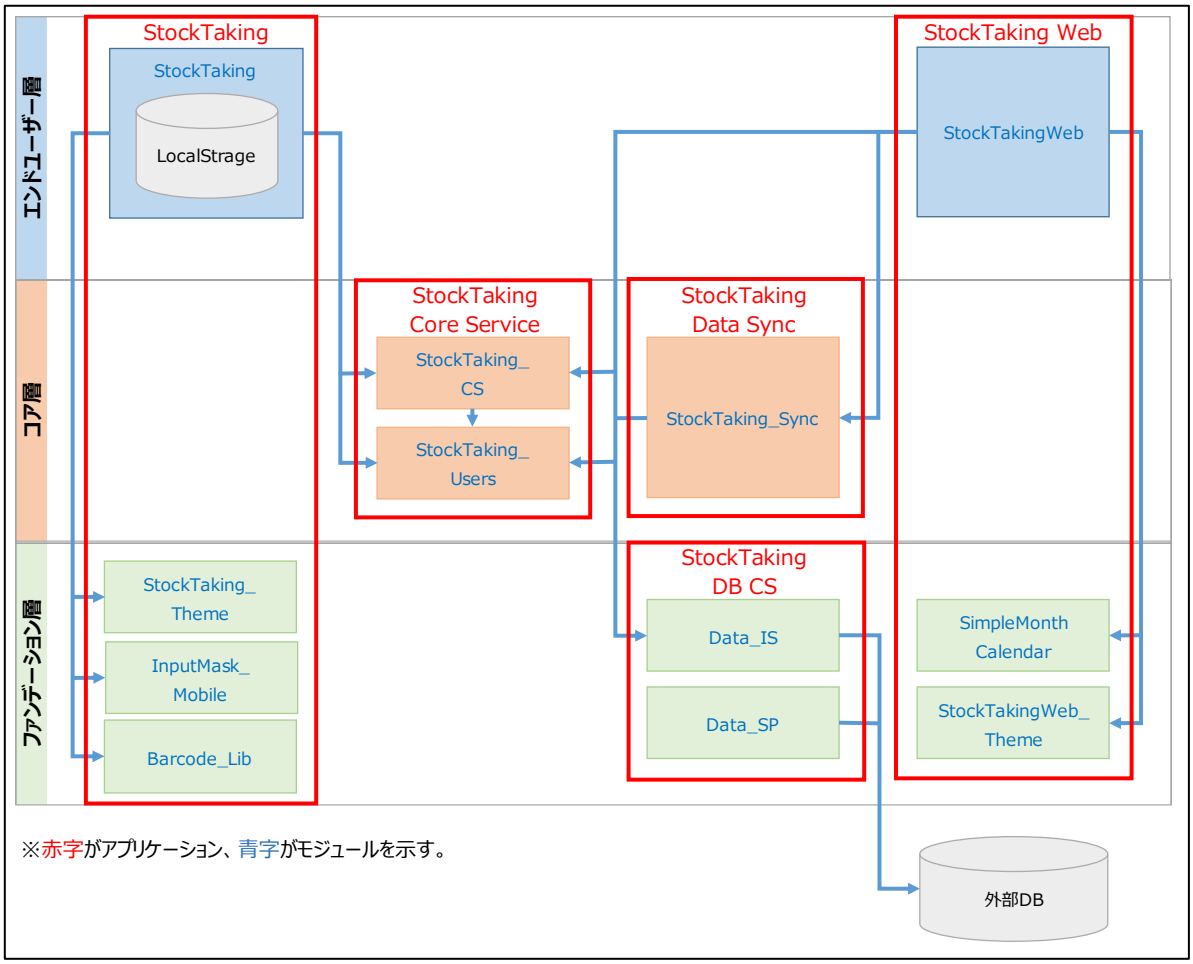


図 6.完成したシステム全体のアーキテクチャ図

終わりに

今回の記事では、OutSystems でアプリケーションを構築する際のベストプラクティスとされている、Architecture Canvas の考え方について、採用事例を交えてご紹介しました。Architecture Canvas のルールに従うことで、SOA の原則に従い、再利用しやすく保守性が高いアーキテクチャを構築することができます。

tdi では、 OutSystems の機能や技術について十分な知識を持った多くの技術者を有しており、資格保有者数は国内トップクラスです。ローコード開発が一般的に注目される以前（2016 年）から重ねた OutSystems 開発の実績をもとに、IT 戦略コンサルティングや OutSystems 導入から運用までをトータルサポートします。また、お客様に合わせた人財育成や内製化もご支援いたします。本件を含め、何かお困りごとがございましたらどうぞお気軽にお問合せください。

【ローコード開発基盤「OutSystems」】

https://www.tdi.co.jp/outsystems/

【お問い合わせ】

https://tdi.smktg.jp/public/application/add/1095

情報技術開発株式会社　営業本部

東京:〒163-1332 東京都新宿区西新宿六丁目 5 番 1 号　新宿アイランドタワー32 階
03-5325-4811 (代表)　FAX 03-5325-4812

中部:〒451-6027 愛知県名古屋市西区牛島町 6 番 1 号 名古屋ルーセントタワー27 階
TEL 052-571-6871（代表）　FAX 052-571-3856

関西:〒530-0005 大阪府大阪市北区中之島二丁目 2 番 7 号 中之島セントラルタワー20 階
TEL.06-6201-7739(代表) FAX.06-6201-7740

九州:〒812-0013 福岡県福岡市博多区博多駅東二丁目 10 番 1 号 福岡ビル S 館 7 階
TEL.092-451-8218(代表) FAX.092-474-7379