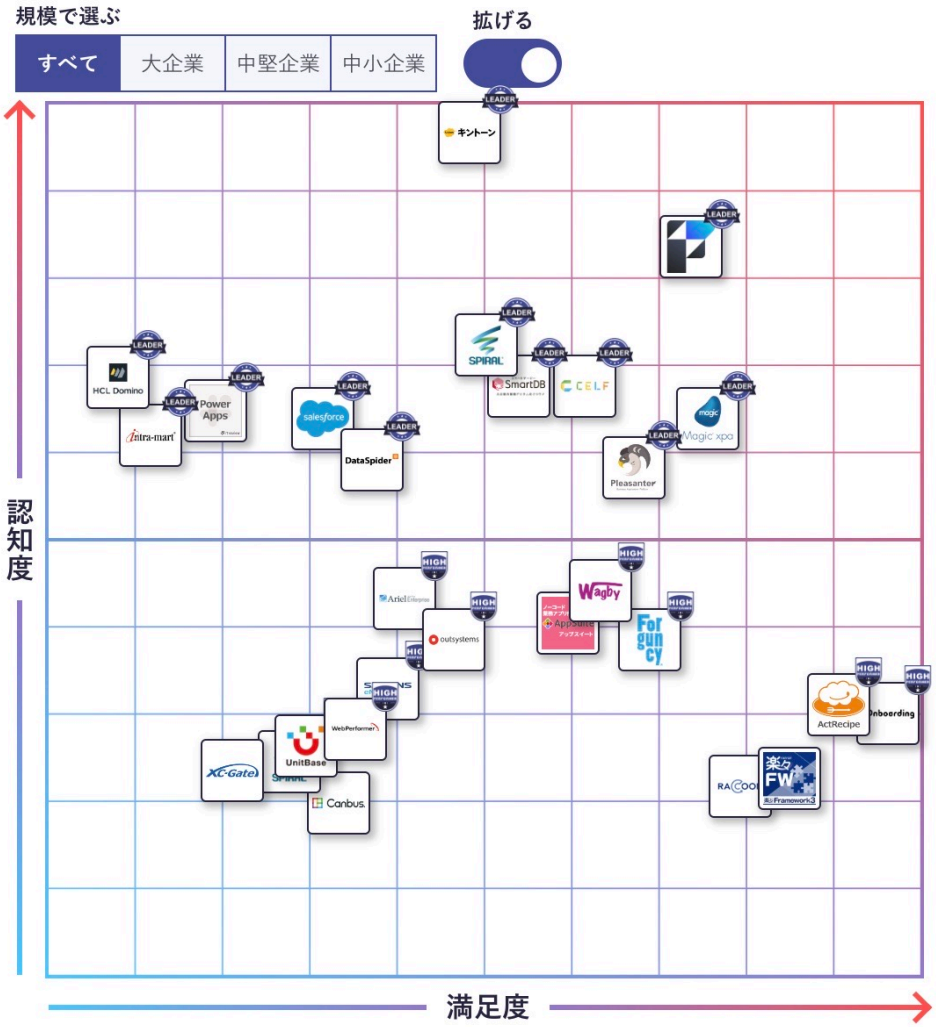
キーノートセッション [改]

株式会社ジャスミンソフト

Wagby Spec リード

贊　良則

https://www.itreview.jp/categories/low-code-development

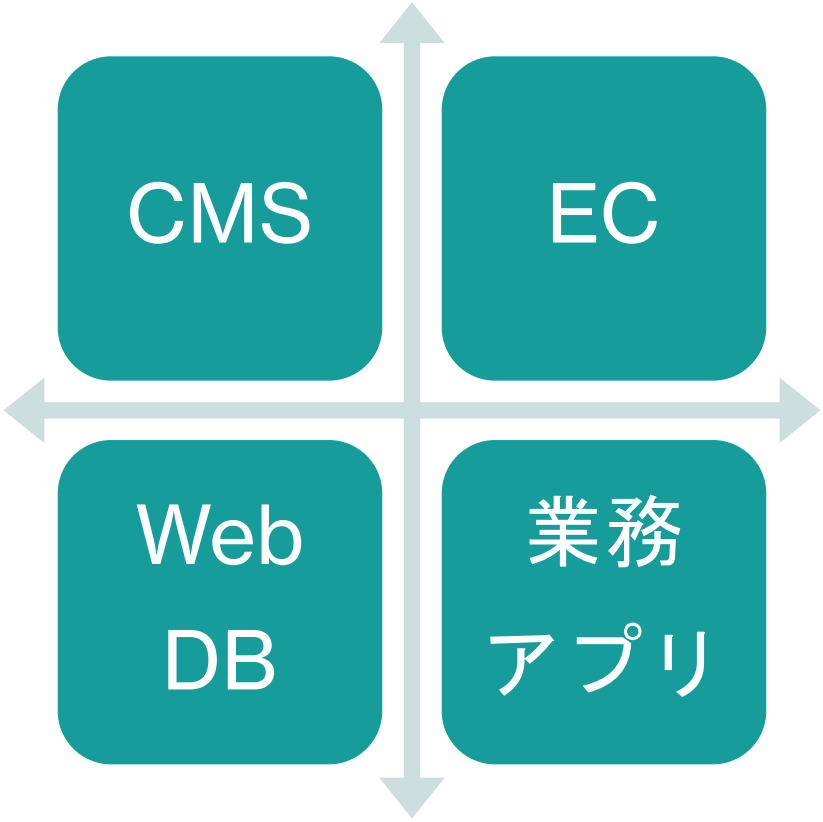


ノーコード、ローコードツール

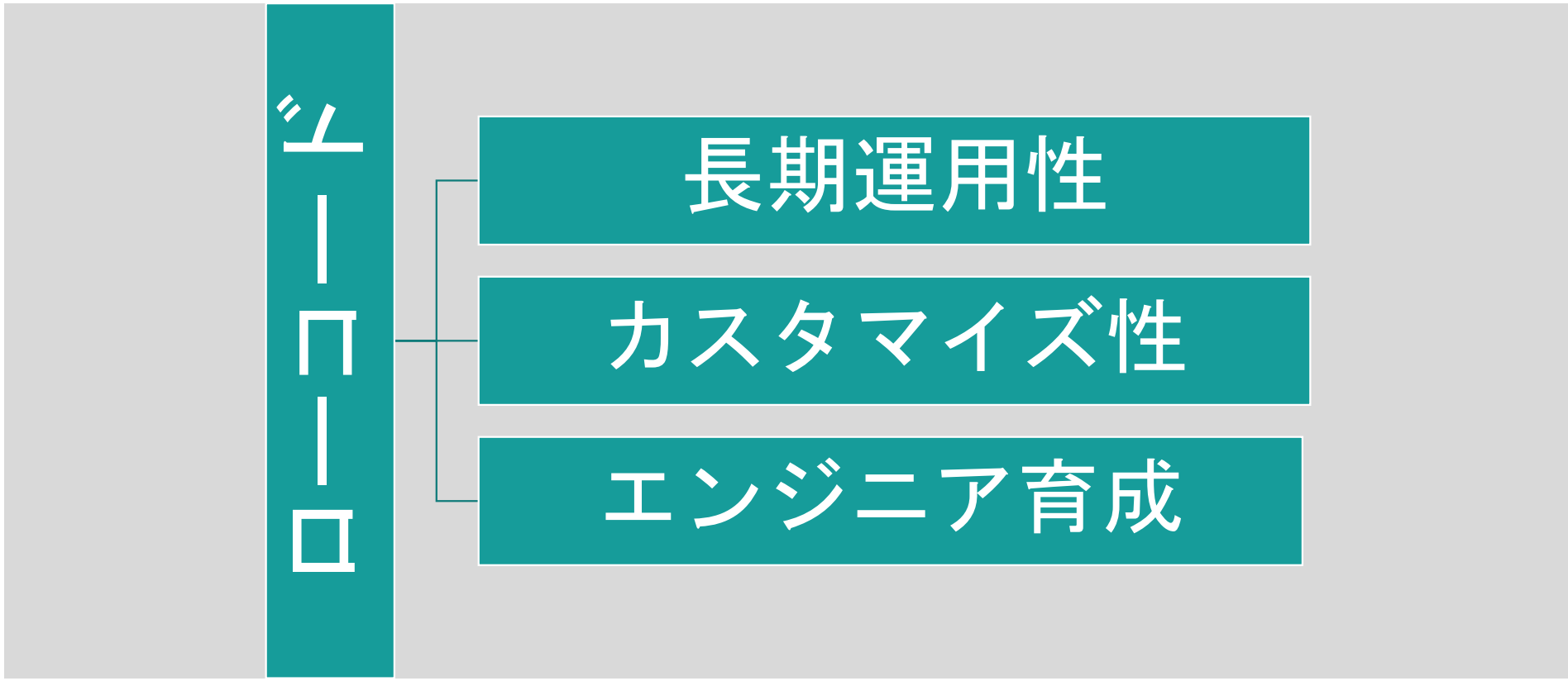
百花繚乱の時代

2023年10月現在、ITreviewで44製品。

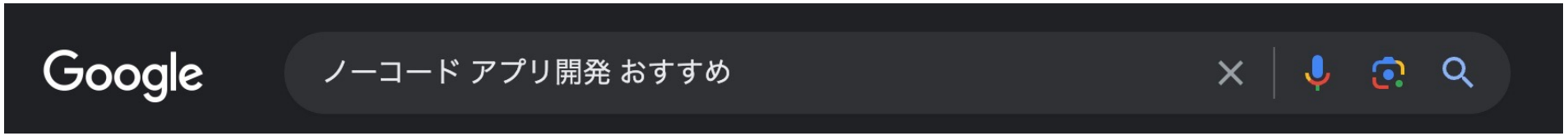
ノーコードは対象が広い（広すぎる）



ローコードへの不安 – 3つの疑問



ノーコードで大量のアプリを作ることが正解か？



オススメ　ノーコードツール10選！特徴・機能・料金について

2023年最新比較　無料のおすすめノーコードツール12選

ノーコードツール比較15選！タイプ別に紹介

ノーコードのアプリが無料で作成できるおすすめツール7選

...

例　資産管理台帳アプリ

PC台帳管理

社用車 台帳管理

社宅 台帳管理

保有物件 台帳管理

郵便切手管理

備品管理

Excelと何が違う？

PC台帳管理

社用車 台帳管理

社宅 台帳管理

保有物件 台帳管理

郵便切手管理

備品管理

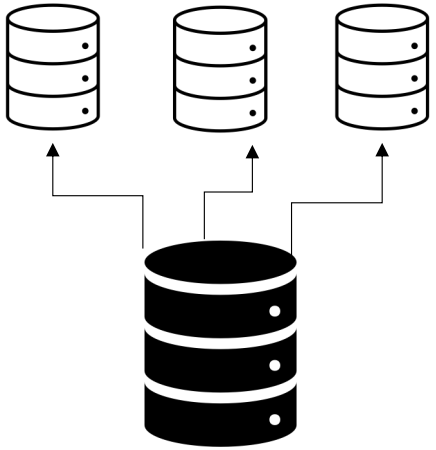
Thanx https://icons8.com

どうすればよかった？

汎用資産管理台帳を設計する

- データ構造を工夫し、さまざまな属性を一元管理できるようにする。
- 単純なノーコードツールでは対応できない。

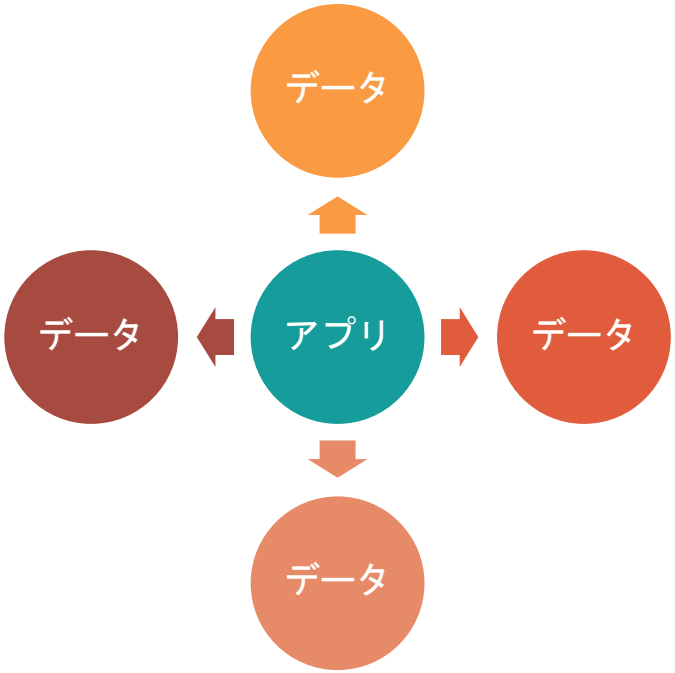
それでも溢れる個別要求に対しては



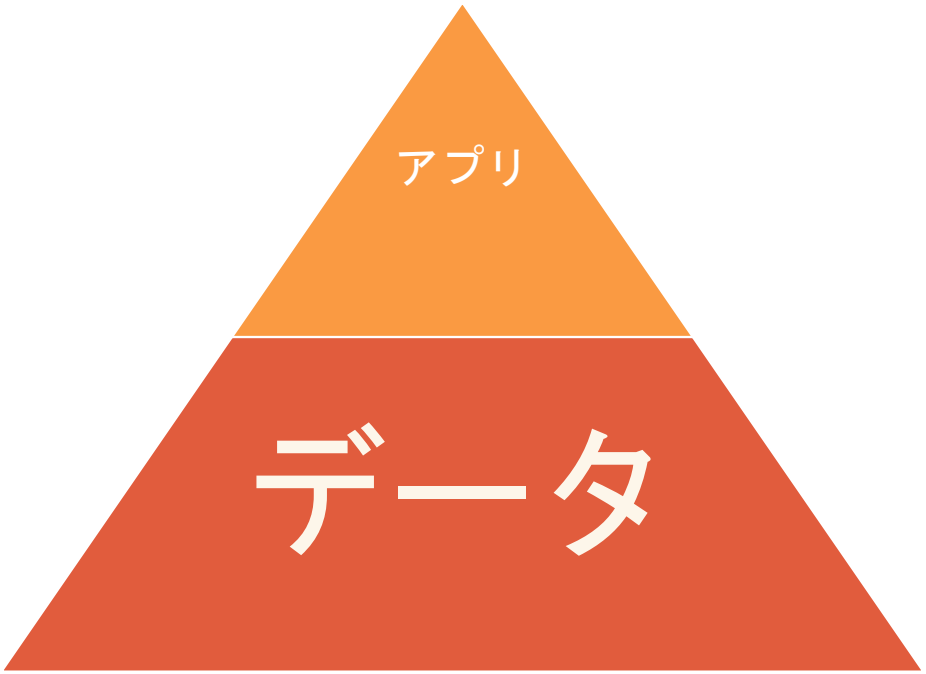
- 共通データ基盤と、個別データ管理アプリの二層に分離する。
- 共通データ基盤とはREST APIなどで疎結合連携する。

データとアプリケーションの関係

プロセス中心設計

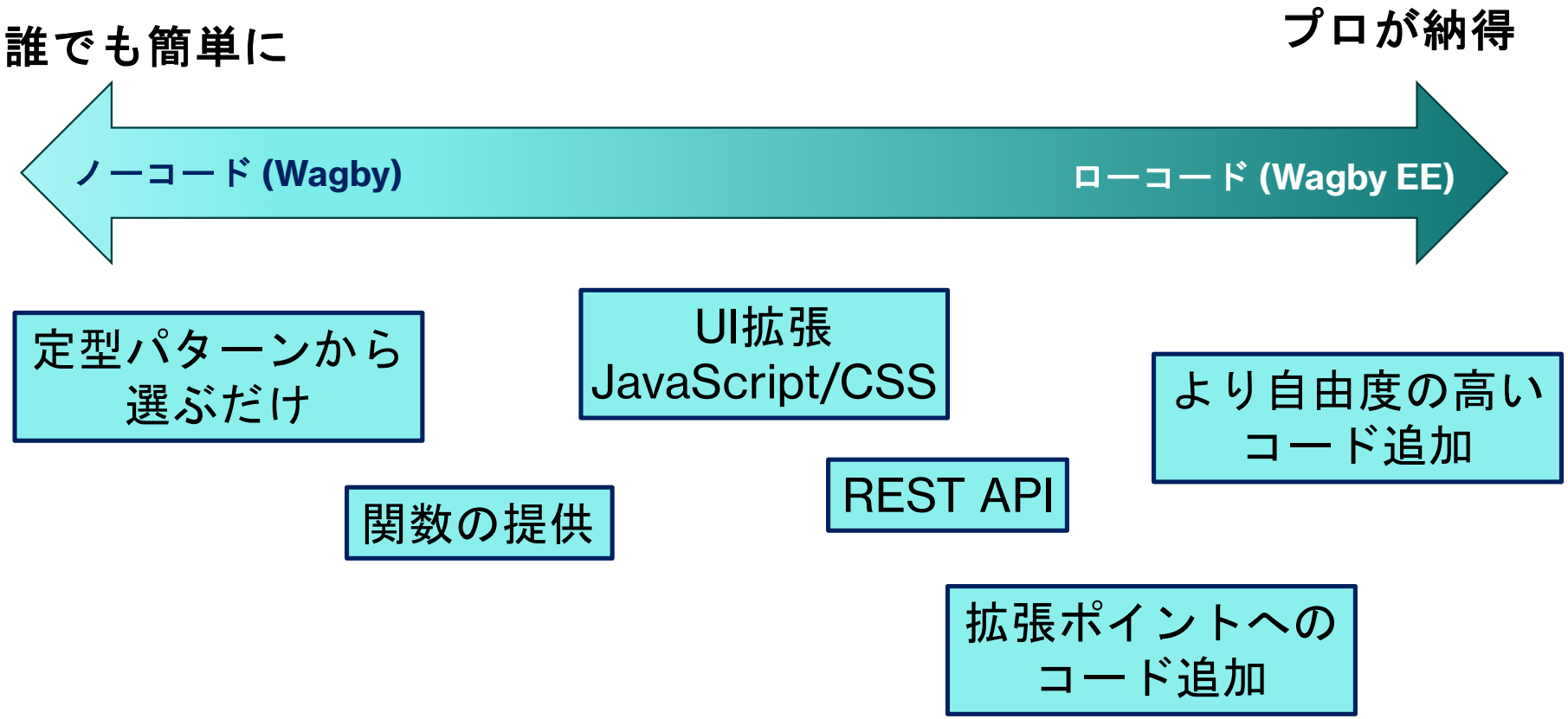


データ中心設計



Wagbyのアプローチ

ノーコード/ローコードツールのカスタマイズ性



業務系で求められるカスタマイズテーマ [1]

入力チェック

必須

文字数

範囲

許容文字（正規表現）

他項目比較

条件付きチェック

他DB項目とのチェック

入力制御

最大入力文字

隠し項目

読み込み専用項目

ルックアップ

他項目値による入力可、不可制御

コピー

条件付きルックアップ

他DB項目と関連した入力制御

トランザクション

同一DB内項目の値を用いた計算

関連する（他DB）項目の同時更新

登録、更新、削除タイミングで他DB操作。任意の条件判断を含む、自由なコード記述。SQL発行を含む。

ストアドプロシージャ呼び出し。

いずれもDBのトランザクションとして完結すること。失敗時のロールバック保証。

業務系で求められるカスタマイズテーマ [2]

入力支援 (UI)

リストボックス、ラジオボタン、チェックボックス、サブウィンドウ検索

入力時はリストボックスだが検索時はチェックボックスにする、など。

サブウィンドウによる複数値の選択

コード直接入力

サジェスト入力

条件付き警告ダイアログ

明細入力 (UI)

明細行の追加、削除、挿入

一行段組レイアウト

最大行の指定

行内の項目、行外の項目を組み合わせた入力可、不可制御

条件付きの行追加、削除、挿入制御

その他 (UI)

郵便番号と住所の連動

カナ自動入力

電話番号ハイフン自動補完

ルビ

グラフ、バーゲージ

バーコード入力、出力

ヘルプ

ポータル画面

独自ボタンによる、オリジナル画面操作

業務系で求められるカスタマイズテーマ [3]

外部連携

PDF 出力

CSV/Excel ダウンロード

CSV/Excel アップロード更新

ダウンロード、アップロード時の値の加工処理

非同期処理

REST API サーバ化

外部REST API呼び出し

非機能要件

複数サーバ化（クラスタリング、オートスケール）

複数データベース対応

国際化

マルチセッション（ブラウザの複数タブ対応）

マルチテナント（SaaSアプリケーション提供として）

その他

既存テーブルとの連携（複合キーテーブルあり）

複数データベースの同時利用

シングルサインオン

Open ID Connect

ログオン、ログオフ時の追加処理

上流工程との連携

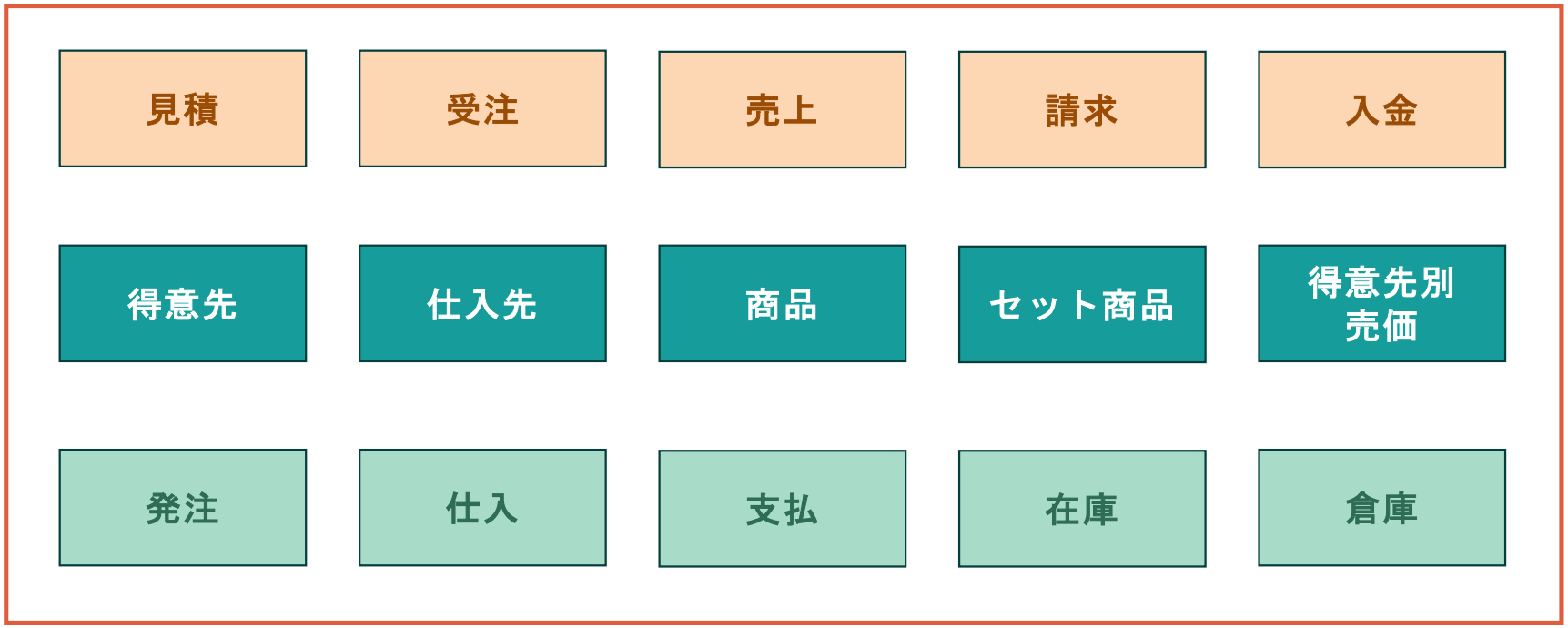
設計書の出力

セキュリティ脆弱性対応

事例

SaaS型販売管理アプリケーション

販売・仕入管理



カスタマイズ方法

- 自動生成されたファイルを直接、変更しない。
- customizeフォルダに「追加」のファイルを配置する。

mycommon.js	common.js を強化する。
<モデルID>/My.js	そのモデルに関わるすべての画面に適用
<モデルID>/My<画面>.js	そのモデルのその画面に適用
<モデルID>/My<画面>.css	そのモデルのその画面に適用

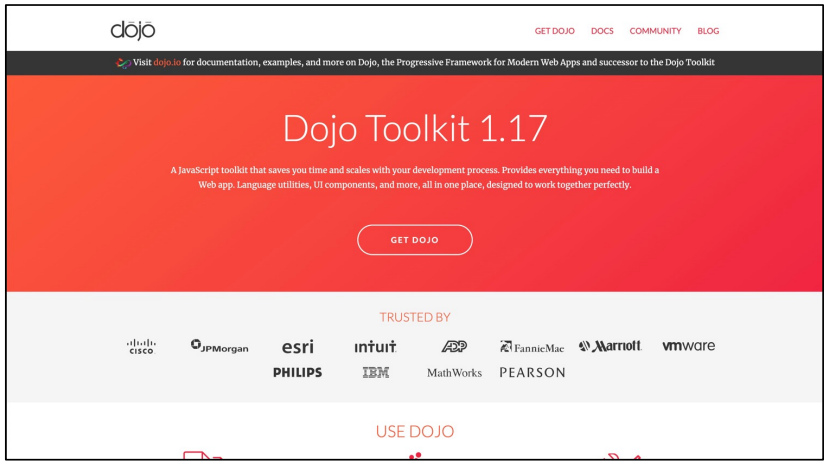
サーバ側（DB操作含む）例

```
/**
 * 在庫情報を更新します(削除レコード用)。
 */
function updateZaikoDeleteTransaction(headerData, detailsData) {
 // 倉庫の指定なしは想定外。
 if (detailsData.getSokocdAsInteger() == null) {
  return;
 }

 var zaikoProcessor = null;
 var juchuSokoZaikoProcessor = null
 var subGenzaizaikosu = function(mstZaiko) {
  // 現在在庫数
  mstZaiko.setGenzaizaikosu(VALUE(ADD(mstZaiko.getGenzaizaikosu(0), detailsData.getSuryo(0))));
 };
```

```
var subJuchuzansu = function(mstZaiko) {
  /** 受注残数 **/
  //出荷指図＝0、=1、=2 の場合のみ
  if ((Constants.SHUKKASASHIZU_ZAIKO_H.intValue() ==  detailsData.getShukkasashizu(-1)
     || Constants.SHUKKASASHIZU_H_TSUJYO.intValue() ==  detailsData.getShukkasashizu(-1)
     || Constants.SHUKKASASHIZU_H_CHOKUSO.intValue() ==  detailsData.getShukkasashizu(-1))) {
    mstZaiko.setJuchuzansu(VALUE(ADD(mstZaiko.getJuchuzansu(0), detailsData.getSuryo(0))));
  }
 };

 var key = new Packages.jp.jasminesoft.wagby.model.mst_zaiko.MstZaiko.Key();
 key.setKanyushabango(detailsData.kanyushabango);
 key.setShohincd(detailsData.getShohincd());
 if (zaikoProcessor != null) {
  key.setSokocd(detailsData.getSokocd());
  updateZaikoByKeyDeleteTransaction(key, detailsData, zaikoProcessor)
 }
}
```

フロントエンド（UI限定・DB操作なし）

- すべての「部品」は Dojotoolkit を用いている。
- その他のライブラリ…例：jQuery…を混ぜることは不可能ではないが、推奨していない。
- Dojotoolkitの作法を学ぶことが、Wagby UI カスタマイズのベストソリューション。



フロントエンド JavaScript 例1

```
// 在庫数照会ダイアログに表示する HTML を返す
 function getZaikoHTML(rowNum) {
return `
<table class="display_table">
 <tbody>
  <tr><th >A在庫数</th><td>${genzaizaikosu}</td></tr>
  <tr><th>B内在庫引当分</th><td>${hikiatezaikosu}</td></tr>
  <tr><th>C出荷可能数(A-B)</th><td>${sukkakanousu}</td></tr>
 </tbody>
</table>
<div style="display: flex; justify-content: center; margin-top: 15px;">
 <button id="btnOpenHacchuzanMeisaiDialog" style="margin: 0 15px;">発注残明細</button>
 <button id="btnOpenTaSokoZaikoDialog" style="margin: 0 15px;">他倉庫の在庫</button>
</div>
`;
 }
```

フロントエンド JavaScript 例2

```
// 与信チェックのダイアログに表示するメッセージを取得する
 function getYoshinCheckMessage(yoshingendogaku, urikakezandaka) {
  return `取引限度額以上の金額が入力されています。<br>このまま登録してもよろしいですか？<br>
<table style="margin: 8px 0 10px 15px;">
 <tbody>
  <tr>
   <td>与信限度額</td><td>:</td><td>${getFormattedNumber(yoshingendogaku)}</td>
  </tr>
  <tr>
   <td>売掛残高</td><td>:</td><td>${getFormattedNumber(urikakezandaka)}</td>
  </tr>
 </tbody>
</table>
`;
 }
```

フロントエンド JavaScript 例3

```
// 発注残明細ダイアログの生成
 function createHacchuzanMeisaiDialog(rowNum) {
  createConfirmDialog({
    "title": "発注明細照会",
    "type": "HacchuzanMeisai",
    "icon": "none",
    "cancelButtonLabel": "閉じる",
    "style": "width: 700px",
    "message": getHacchuzanMeisaiHTML(rowNum),
  }, function(dialog) {
    setDialogStyle(dialog);
    dialog.show();
    wbHacchuzanMeisaiDialog = dialog;
  });
 }
```

フロントエンド JavaScript 例4

```
function createBtnHacchuzanMeisai(rowNum) {// 「発注残明細」ボタンを配置する
 var btnHacchuzanMeisai = registry.byId("btnOpenHacchuzanMeisaiDialog");
 if (btnHacchuzanMeisai) {
  btnHacchuzanMeisai.destroy();
 }
 btnHacchuzanMeisai = new Button({
 }, dom.byId("btnOpenHacchuzanMeisaiDialog"));
 btnHacchuzanMeisai.on("click", function() {
  if (!wbHacchuzanMeisaiDialog) {// 発注残明細ダイアログが未生成の場合は新規作成する
   createHacchuzanMeisaiDialog(rowNum);
  } else {// 発注残明細ダイアログが生成済の場合は出力内容を更新する
   wbHacchuzanMeisaiDialog.updateMessage(getHacchuzanMeisaiHTML(rowNum));
   wbHacchuzanMeisaiDialog.show();
  }
 });
}
```

フロントエンド CSS 例

```
.sempotantoshamei_label,
.chokusotantoshamei_label,
.biko_label {
 width: 180px;
}
.sempotantoshamei_field,
.chokusotantoshamei_field,
.biko_field {
  width: unset;
}
/* 金額項目を右寄せにするため width:unset; は利用不可 */
.arariekiritsu_field,
.ararieki_field {
width:calc(100% - 205px);
}
```

カスタマイズで、できること

- 複雑な入力チェック
- 複雑な入力制御
- 複雑なデータベース検索、更新処理
- データベースビュー、トリガーとの組み合わせ
- データベースストアドプロシージャ呼び出し
- 自動生成された処理の改変（基本動作の個別修正）
- UI 操作：ボタン配置、ダイアログ配置、レイアウト修正

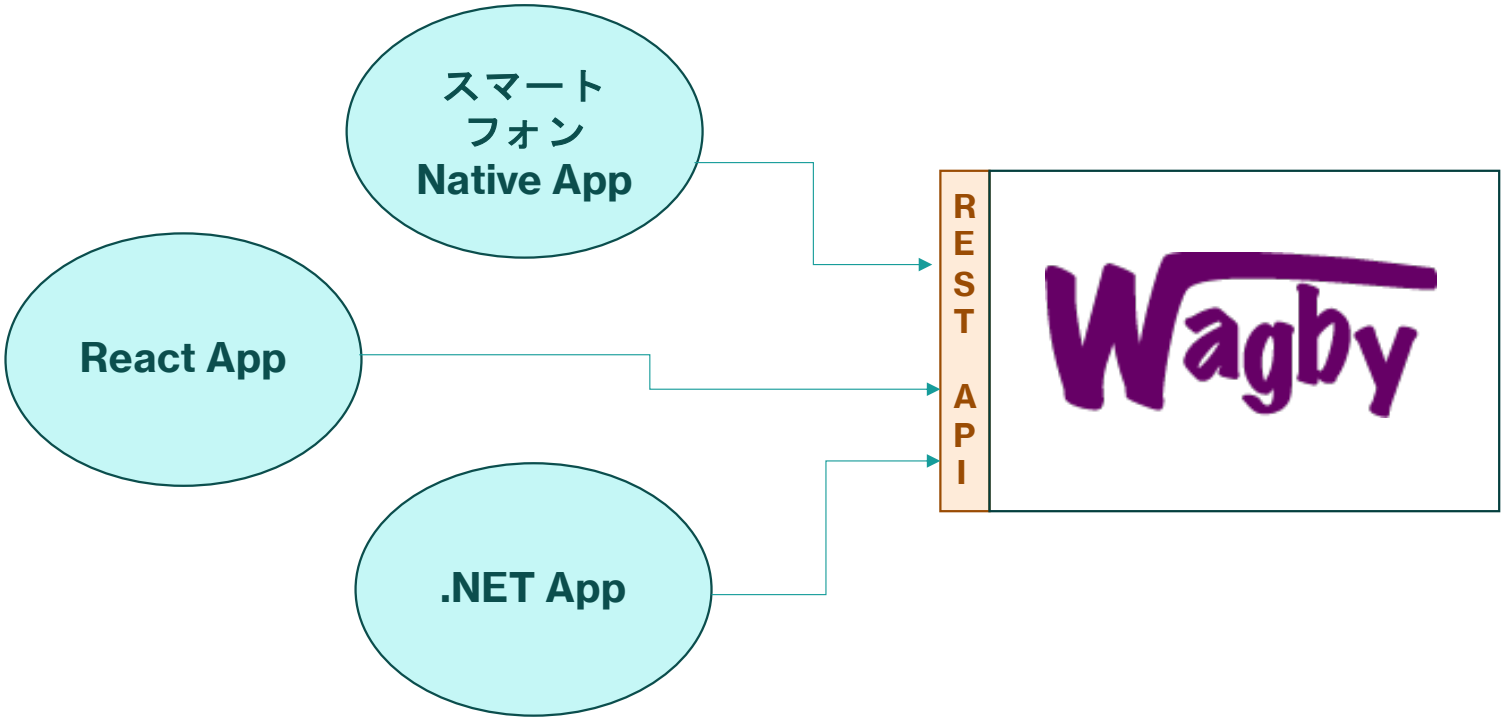
これまで蓄積されてきたJava
クラスライブラリ、データ
ベーステーブルの再利用も。

カスタマイズでも、できないこと

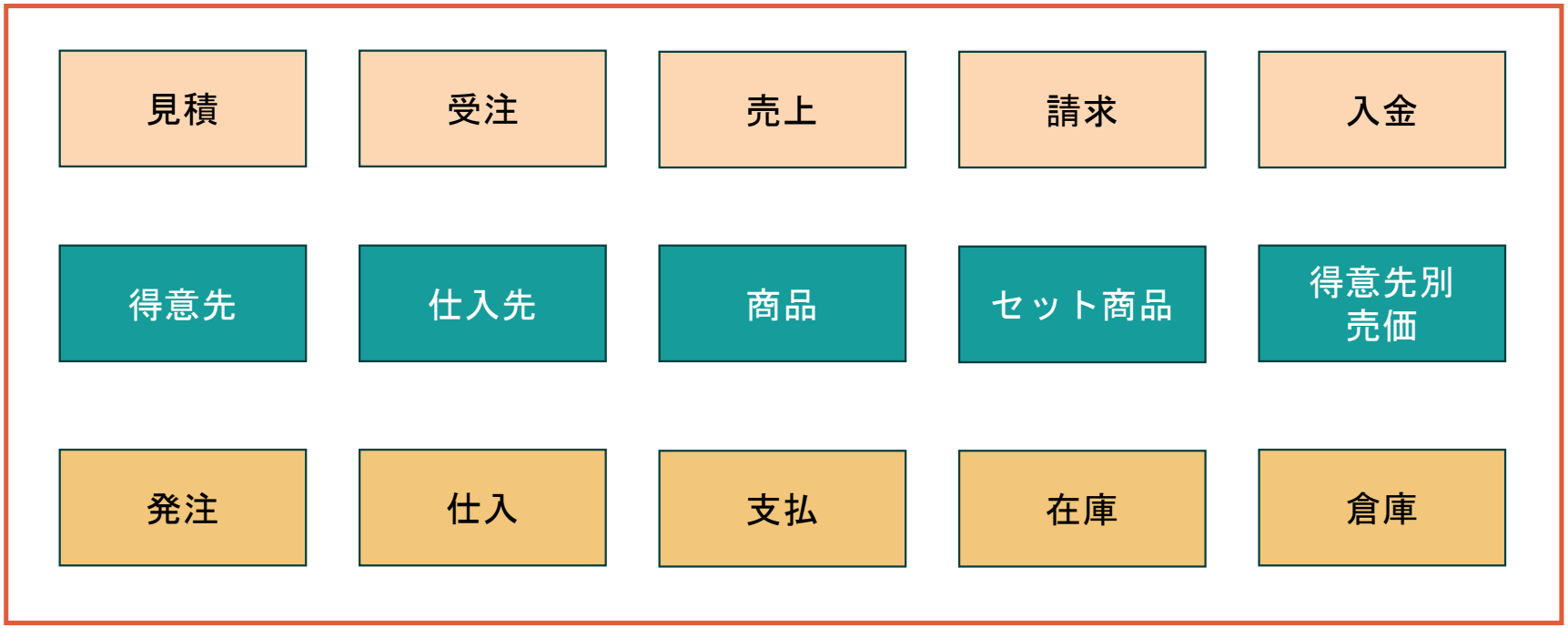
- 認証、認可
  - Wagbyが利用している Spring Security を前提としたカスタマイズは可能だが、Spring Security を変えることはできない。
- Wagbyが対応していないデータベースの利用
  - JDBC 4 規格が必要。
- UIの大幅な変更
  - WagbyがCSS/JavaScriptとは異なるアーキテクチャは利用できない。

回避策 REST API サーバ

独自フロントエンドから、WagbyのREST APIを呼び出す。

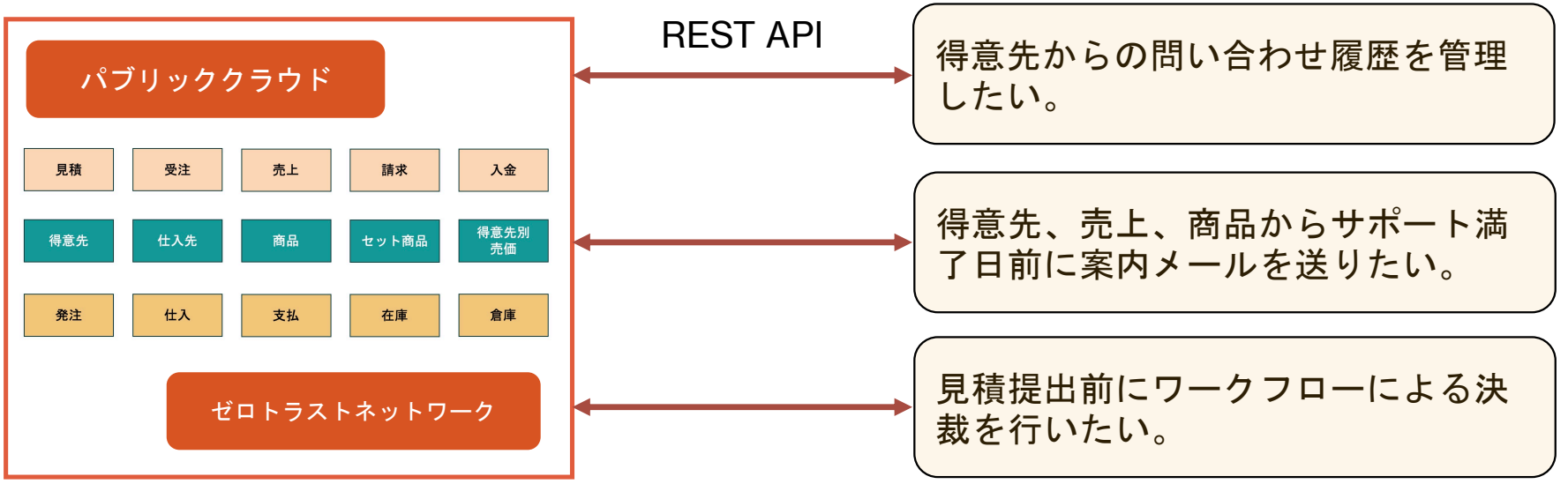


データ中心設計を支えるためのローコード開発



例　販売・仕入管理

基幹データを有効活用するノーコード開発



Wagby EE で開発する
基幹業務アプリケーション

Wagby で開発する
ノーコードアプリケーション

簡単なWebアプリの開発から、本格的な業務システムの開発まで

利用した開発担当者が「もう手放せない」と口コミする
知る人ぞ知る“一挙両得”のローコード開発ツールとは？

Wagby

Powered By ITreview

October 2023

簡単な開発でも、基幹系の開発でも「Wagbyは一挙両得のツール」と語るレビュー

Pickup!

研究
ソフトウェア・SI
1000人以上

★★★★☆

お手軽・簡単なものから基幹系まで対応できる製品

多くの設定項目があり、さらにサーバーサイドのロジックをコーディングできるので、かなりの作り込みが可能です。**完全ノンプログラミングのお手軽・簡単なものから、しっかりと制御を利かせたいものまで**幅広く対象にできます。**基幹システムにも対応できるポテンシャルを持っている**と感じます。社内や部内で利用するアプリケーション開発から始めました。生産性の向上は確実に感じられました。自動生成により、**製造工数は大幅に削減できる**と思います。また、自動生成される部分（＝人が作っていない部分）は単体テストレベルのバグが発生しないため、**テスト工数の削減**にもなります。

https://www.itreview.jp/products/wagby/reviews/56711

Pickup!

その他専門職
ソフトウェア・SI
1000人以上

★★★★☆

小規模な部門内アプリからSI含めた基幹システム開発までカバー

Wagbyはユーザー側主体で**部門内＆チーム内での活用から中規模〜大規模な基幹システム開発の範囲までカバー＆連携できる点が強み**だと思います。小規模〜中規模の案件においてWagbyがカバーできる範囲でノンカスタマイズで活用するのが最も効果が高いと思います。一方で**中規模以上の基幹システムでも活用可能**。Wagbyのアーキテクチャーが Java/Spring/Hibernate といったオープンなWeb標準技術で実現されており、SIer/エンジニアにとって理解しやすく、部分的に活用したり連携することが行いやすいからだと感じています。うまく活用すれば、**納期と保守性の面で大きな効果**が出せました。

https://www.itreview.jp/products/wagby/reviews/56675

Pickup!

社内情報システム（開発・運用）
その他の化学工業
300-1000人未満

★★★½☆

Web業務システムを超高速で作る

簡易な登録・更新・検索・閲覧の画面機能であれば、30分程度で動く画面を開発できます。既存システムのテーブル利用が可能で、基幹システム周辺の個別システムをスモールパッケージで多数開発することにも適しています。**複雑なビジネスロジックが必要な場合、サーバサイドスクリプトが大変便利**です。ECMAScriptによる**サーバサードロジックがビルド不要でアプリ稼働中でも更新可能な点が重宝**しています。作成されたソースがJava・SpringFrameworkベースであり、生成されたソースも可視性が高く、**JavaWeb開発者にとっても順応性・拡張性が見込めます。**

https://www.itreview.jp/products/wagby/reviews/56452

最後まで

お楽しみ

ください！